# Bluetooth Mesh Software API Reference Manual

This document contains the full API reference for the Silicon Labs Bluetooth Mesh Software version 1.7.5, based on the Bluetooth LE Software version 2.13.10.

The Blue Gecko family of the Silicon Labs' Bluetooth chipsets deliver a high performance, low energy and easy-to-use Bluetooth solution integrated into a small form factor package.

The ultra-low power operating modes and fast wake-up times of the Silicon Labs' energy friendly 32-bit MCUs, combined with the low transmit and receive power consumption of the Bluetooth radio, result in a solution optimized for battery powered applications.

# Table of Contents

# 1. Data types

Data types used in the documentation are shown in the table below. Unless otherwise noted, all multi-byte fields are in little endian format.

**Table 1.1. Data types**

| Name | Length | Description |
|---|---|---|
| errorcode | 2 bytes | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| int16 | 2 bytes | Signed 16-bit integer |
| bd_addr | 6 bytes | Bluetooth address |
| uint16 | 2 bytes | Unsigned 16-bit integer |
| int32 | 4 bytes | Signed 32-bit integer |
| uint32 | 4 bytes | Unsigned 32-bit integer |
| link_id_t | 2 bytes | Link ID |
| int8 | 1 byte | Signed 8-bit integer |
| uint8 | 1 byte | Unsigned 8-bit integer |
| uint8array | 1 - 256 bytes | Variable length byte array. The first byte defines the length of data that follows, 0 - 255 bytes. |
| ser_name | 16 bytes | Service name, 16-byte array |
| dbm | 1 byte | Signal strength |
| connection | 1 byte | Connection handle |
| service | 4 bytes | GATT service handle<br>This value is normally received from the gatt_service event. |
| characteristic | 2 bytes | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| descriptor | 2 bytes | GATT characteristic descriptor handle |
| uuid | 3 or 17 bytes | uint8array containing a 2 or 16-byte Universal Unique Identifier (UUID) |
| att_errorcode | 1 byte | Attribute protocol error code<br>• **0:** No error<br>• **Non-zero:** See Bluetooth specification, Host volume, Attribute Protocol, Error Codes table. |
| att_opcode | 1 byte | Attribute opcode that informs the procedure from which the value was received. |
| uuid_128 | 16 bytes | 128-bit UUID |
| aes_key_128 | 16 bytes | 128-bit AES Key |
| uuid_64 | 8 bytes | 64-bit UUID |
| int64 | 8 bytes | Signed 64-bit integer |
| uint64 | 8 bytes | Unsigned 64-bit integer |

# 2. API Reference

This section describes all commands, enumerations, responses, events and errors. Commands with related enumerations, responses and events are grouped according to command classes.

### BGAPI Payload

The parameters of a BGAPI command, response, or event are passed between the application and firmware in a payload. For example, a parameter of uint32 type uses 4 bytes of the payload space. A byte array parameter uses one byte to describe the length of the array. Data in the array is copied into the remaining free payload space.

### Maximum BGAPI Payload Size

The maximum BGAPI payload size is 256 bytes for both NCP and SoC modes. When an application calls a BGAPI command, BGAPI checks the payload length and returns an error code 0x018a (command_too_long) if the payload causes an overflow.

### Deprecation Notice

Note that some commands, enumerations, and events are marked as deprecated. Avoid using those commands because they will be removed in future releases.

### Sensitive Data Handling

Certain commands in the Mesh classes read or write security-critical material. In Secure NCP applications, the BGAPI communication between the host and the target must be encrypted. Otherwise, the commands will return the error code 0x0a0e mismatched_or_insufficient_security. This feature does not affect SoC or non-secure NCP applications.

## 2.1 Coexistence Interface (coex)

Coexistence BGAPI class. Coexistence interface is enabled and initialized with gecko_initCoexHAL() function.

### 2.1.1 coex commands

#### 2.1.1.1 cmd_coex_get_counters

Read coexistence statistic counters from the device. Response contains the list of uint32 type counter values. Counters in the list are in following order: low priority requested, high priority requested, low priority denied, high priority denied, low-priority TX aborted, and high-priority TX aborted. Passing a non-zero value also resets counters.

**Table 2.1. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | reset | Reset counters if parameter value is not zero. |

**Table 2.2. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | counters | Coexistence statistic counters |

**BGLIB C API**

```
/* Function */
struct gecko_msg_coex_get_counters_rsp_t *gecko_cmd_coex_get_counters(uint8 reset);

/* Response id */
gecko_rsp_coex_get_counters_id

/* Response structure */
struct gecko_msg_coex_get_counters_rsp_t
{
  uint16 result;,
  uint8array counters;
};
```

#### 2.1.1.2 cmd_coex_set_directional_priority_pulse

Set Directional Priority Pulse Width

**Table 2.3. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | pulse | Directional priority pulse width in us |

**Table 2.4. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_coex_set_directional_priority_pulse_rsp_t
*gecko_cmd_coex_set_directional_priority_pulse(uint8 pulse);

/* Response id */
gecko_rsp_coex_set_directional_priority_pulse_id

/* Response structure */
struct gecko_msg_coex_set_directional_priority_pulse_rsp_t
{
  uint16 result;
};
```

#### 2.1.1.3 cmd_coex_set_options

Configure coexistence options at runtime.

**Table 2.5. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | mask | Mask defines which coexistence options are changed. |
| 8-11 | uint32 | options | Value of options to be changed. This parameter is used together with the mask parameter. |

**Table 2.6. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_coex_set_options_rsp_t *gecko_cmd_coex_set_options(uint32 mask, uint32 options);

/* Response id */
gecko_rsp_coex_set_options_id

/* Response structure */
struct gecko_msg_coex_set_options_rsp_t
{
  uint16 result;
};
```

#### 2.1.1.4 cmd_coex_set_parameters

Configure coexistence parameters.

**Table 2.7. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | priority | Coexistence priority threshold. Coexistence priority is toggled if priority is below this value. |
| 5 | uint8 | request | Coexistence request threshold. Coexistence request is toggled if priority is below this value. |
| 6 | uint8 | pwm_period | PWM functionality period length in 1ms units |
| 7 | uint8 | pwm_dutycycle | PWM functionality dutycycle in percentage |

**Table 2.8. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x20 | class | Message class: Coexistence Interface |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_coex_set_parameters_rsp_t *gecko_cmd_coex_set_parameters(uint8 priority, uint8 request, uint8
pwm_period, uint8 pwm_dutycycle);

/* Response id */
gecko_rsp_coex_set_parameters_id

/* Response structure */
struct gecko_msg_coex_set_parameters_rsp_t
{
  uint16 result;
};
```

### 2.1.2 coex enumerations

#### 2.1.2.1 enum_coex_option

Coexistence configuration options

**Table 2.9. Enumerations**

| Value | Name | Description |
|---|---|---|
| 256 | coex_option_enable | Enable coexistence feature |
| 1024 | coex_option_tx_abort | Abort transmission if grant is denied |
| 2048 | coex_option_high_priority | Enable priority signal |

## 2.2 CTE Receiver (cte_receiver)

Commands and events in this class manage Constant Tone Extension (CTE) receiving.

CTE feature is only supported by specific devices. Commands from this class will return bg_err_not_supported on devices that do not support CTE.

### 2.2.1 cte_receiver commands

#### 2.2.1.1 cmd_cte_receiver_clear_dtm_parameters

Clear CTE-related parameters that were previously set for LE receiver test. Default values will be restored for these parameters.

**Table 2.10. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x06 | method | Message ID |

**Table 2.11. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_cte_receiver_clear_dtm_parameters_rsp_t *gecko_cmd_cte_receiver_clear_dtm_parameters();

/* Response id */
gecko_rsp_cte_receiver_clear_dtm_parameters_id

/* Response structure */
struct gecko_msg_cte_receiver_clear_dtm_parameters_rsp_t
{
  uint16 result;
};
```

#### 2.2.1.2 cmd_cte_receiver_disable_connection_cte

Stop the IQ sampling on a connection. CTEs will not be requested on the given connection.

**Table 2.12. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.13. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_cte_receiver_disable_connection_cte_rsp_t
*gecko_cmd_cte_receiver_disable_connection_cte(uint8 connection);

/* Response id */
gecko_rsp_cte_receiver_disable_connection_cte_id

/* Response structure */
struct gecko_msg_cte_receiver_disable_connection_cte_rsp_t
{
  uint16 result;
};
```

#### 2.2.1.3 cmd_cte_receiver_disable_connectionless_cte

Stop IQ sampling on a periodic advertising synchronization.

**Table 2.14. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | sync | Periodic advertising synchronization handle |

**Table 2.15. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_cte_receiver_disable_connectionless_cte_rsp_t
*gecko_cmd_cte_receiver_disable_connectionless_cte(uint8 sync);

/* Response id */
gecko_rsp_cte_receiver_disable_connectionless_cte_id

/* Response structure */
struct gecko_msg_cte_receiver_disable_connectionless_cte_rsp_t
{
  uint16 result;
};
```

#### 2.2.1.4 cmd_cte_receiver_disable_silabs_cte

Disable IQ sampling of Silicon Labs CTE.

**Table 2.16. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x08 | method | Message ID |

**Table 2.17. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_cte_receiver_disable_silabs_cte_rsp_t *gecko_cmd_cte_receiver_disable_silabs_cte();

/* Response id */
gecko_rsp_cte_receiver_disable_silabs_cte_id

/* Response structure */
struct gecko_msg_cte_receiver_disable_silabs_cte_rsp_t
{
  uint16 result;
};
```

#### 2.2.1.5 cmd_cte_receiver_enable_connection_cte

Start IQ samplings on a connection. A CTE requests will be initiated periodically on the given connection and IQ sampling will be made on the received CTE responses.

**Table 2.18. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | interval | Measurement interval<br>• **0:** No interval. The request is initiated only once.<br>• **Other values N:** Initiate the request every N-th connection events |
| 7 | uint8 | cte_length | Minimum CTE length requested in 8 us units.<br>• Range: 0x02 to 0x14<br>• Time Range: 16 us to 160 us |
| 8 | uint8 | cte_type | Requested CTE type<br>• **0:** AoA CTE<br>• **1:** AoD CTE with 1 us slots<br>• **2:** AoD CTE with 2 us slots |
| 9 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 10 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. |

**Table 2.19. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_cte_receiver_enable_connection_cte_rsp_t *gecko_cmd_cte_receiver_enable_connection_cte(uint8 connection, uint16 interval, uint8 cte_length, uint8 cte_type, uint8 slot_durations, uint8 switching_pattern_len, const uint8 *switching_pattern_data);
```

```
/* Response id */
gecko_rsp_cte_receiver_enable_connection_cte_id

/* Response structure */
struct gecko_msg_cte_receiver_enable_connection_cte_rsp_t
{
  uint16 result;
};
```

**Table 2.20. Events Generated**

| Event | Description |
|---|---|
| cte_receiver_connection_iq_report | Triggered when IQ samples have been received. |

#### 2.2.1.6 cmd_cte_receiver_enable_connectionless_cte

Start IQ sampling on a periodic advertising synchronization. IQ samples are taken on each CTE found in the periodic advertisements.

**Table 2.21. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | sync | Periodic advertising synchronization handle |
| 5 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 6 | uint8 | cte_count | • **0:** Sample and report all available CTEs<br>• **Other values:** Maximum number of sampled CTEs in each periodic advertising interval |
| 7 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. |

**Table 2.22. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_cte_receiver_enable_connectionless_cte_rsp_t
*gecko_cmd_cte_receiver_enable_connectionless_cte(uint8  sync,  uint8  slot_durations,  uint8  cte_count,  uint8
switching_pattern_len, const uint8 *switching_pattern_data);

/* Response id */
gecko_rsp_cte_receiver_enable_connectionless_cte_id

/* Response structure */
struct gecko_msg_cte_receiver_enable_connectionless_cte_rsp_t
{
  uint16 result;
};
```

**Table 2.23. Events Generated**

| Event | Description |
|---|---|
| cte_receiver_connectionless_iq_report | Triggered when IQ samples have been received. |

#### 2.2.1.7 cmd_cte_receiver_enable_silabs_cte

Enable IQ sampling of Silicon Labs CTE found in extended advertisements.

**Table 2.24. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 5 | uint8 | cte_count | • **0:** Sample and report all available CTEs<br>• **Other values:** Maximum number of sampled CTEs in each extended advertising interval |
| 6 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. |

**Table 2.25. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_cte_receiver_enable_silabs_cte_rsp_t    *gecko_cmd_cte_receiver_enable_silabs_cte(uint8
slot_durations, uint8 cte_count, uint8 switching_pattern_len, const uint8 *switching_pattern_data);

/* Response id */
gecko_rsp_cte_receiver_enable_silabs_cte_id

/* Response structure */
struct gecko_msg_cte_receiver_enable_silabs_cte_rsp_t
{
  uint16 result;
};
```

**Table 2.26. Events Generated**

| Event | Description |
|---|---|
| cte_receiver_silabs_iq_report | Triggered when IQ samples of Silicon Labs CTE have been received. |

#### 2.2.1.8 cmd_cte_receiver_set_dtm_parameters

Set CTE-related parameters of LE receiver test.

**Table 2.27. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | cte_length | Expected CTE length in 8 us units<br>• **0:** No CTE<br>• **0x02 to 0x14:** Expected CTE length<br>Default: 0 (no CTE) |
| 5 | uint8 | cte_type | Expected CTE type<br>• **0:** Expect AoA CTE<br>• **1:** Expect AoD CTE with 1 us slots<br>• **2:** Expect AoD CTE with 2 us slots<br>Default: 0 |
| 6 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each<br>Default: 1 |
| 7 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. Default: empty array |

**Table 2.28. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_cte_receiver_set_dtm_parameters_rsp_t  *gecko_cmd_cte_receiver_set_dtm_parameters(uint8
cte_length,  uint8  cte_type,  uint8  slot_durations,  uint8  switching_pattern_len,  const  uint8
*switching_pattern_data);

/* Response id */
gecko_rsp_cte_receiver_set_dtm_parameters_id

/* Response structure */
struct gecko_msg_cte_receiver_set_dtm_parameters_rsp_t
```

```
{
  uint16 result;
};
```

**Table 2.29. Events Generated**

| Event | Description |
|---|---|
| cte_receiver_dtm_iq_report | Triggered when IQ samples have been received. |

### 2.2.2 cte_receiver events

#### 2.2.2.1 evt_cte_receiver_connection_iq_report

IQ sample report from connection CTE packets.

**Table 2.30. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | status | Status of CTE IQ sampling |
| 6 | uint8 | connection | Connection handle or periodic advertising synchronization handle |
| 7 | uint8 | phy | The PHY on which the packet is received.<br>• **1:** 1M PHY<br>• **2:** 2M PHY |
| 8 | uint8 | channel | The channel on which the CTE packet was received |
| 9 | int8 | rssi | RSSI in the received CTE packet. Unit: dBm |
| 10 | uint8 | rssi_antenna_id | The ID of the antenna on which RSSI was measured |
| 11 | uint8 | cte_type | The CTE type<br>• **0:** AoA CTE response<br>• **1:** AoD CTE response with 1us slots<br>• **2:** AoD CTE response with 2us slots |
| 12 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 13-14 | uint16 | event_counter | The event counter of the connection |
| 15 | uint8array | samples | IQ samples of the received CTE packet. I and Q samples follow each other alternately (I, Q, I, Q , ...) |

**C Functions**

```
/* Event id */
gecko_evt_cte_receiver_connection_iq_report_id

/* Event structure */
struct gecko_msg_cte_receiver_connection_iq_report_evt_t
{
  uint16 status;,
  uint8 connection;,
  uint8 phy;,
  uint8 channel;,
  int8 rssi;,
  uint8 rssi_antenna_id;,
  uint8 cte_type;,
  uint8 slot_durations;,
  uint16 event_counter;,
  uint8array samples;
};
```

#### 2.2.2.2 evt_cte_receiver_connectionless_iq_report

IQ sample report from connectionless CTE packets.

**Table 2.31. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | status | Status of CTE IQ sampling |
| 6 | uint8 | sync | Periodic advertising synchronization handle |
| 7 | uint8 | channel | The channel on which the CTE packet was received |
| 8 | int8 | rssi | RSSI in the received CTE packet. Unit: dBm |
| 9 | uint8 | rssi_antenna_id | The ID of the antenna on which RSSI was measured |
| 10 | uint8 | cte_type | The CTE type<br>• **0:** AoA CTE response<br>• **1:** AoD CTE response with 1us slots<br>• **2:** AoD CTE response with 2us slots |
| 11 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 12-13 | uint16 | event_counter | The event counter of the periodic advertising train |
| 14 | uint8array | samples | IQ samples of the received CTE packet. I and Q samples follow each other alternately (I, Q, I, Q , ...) |

**C Functions**

```
/* Event id */
gecko_evt_cte_receiver_connectionless_iq_report_id

/* Event structure */
struct gecko_msg_cte_receiver_connectionless_iq_report_evt_t
{
  uint16 status;,
  uint8 sync;,
  uint8 channel;,
  int8 rssi;,
  uint8 rssi_antenna_id;,
  uint8 cte_type;,
  uint8 slot_durations;,
  uint16 event_counter;,
  uint8array samples;
};
```

#### 2.2.2.3 evt_cte_receiver_dtm_iq_report

IQ sample report from DTM CTE packets.

**Table 2.32. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | status | Status of CTE IQ sampling |
| 6 | uint8 | channel | The channel on which the CTE packet was received |
| 7 | int8 | rssi | RSSI in the received CTE packet. Unit: dBm |
| 8 | uint8 | rssi_antenna_id | The ID of the antenna on which RSSI was measured |
| 9 | uint8 | cte_type | The CTE type<br>• **0:** AoA CTE response<br>• **1:** AoD CTE response with 1us slots<br>• **2:** AoD CTE response with 2us slots |
| 10 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 11-12 | uint16 | event_counter | The event counter of the periodic advertising train or the connection |
| 13 | uint8array | samples | IQ samples of the received CTE packet. I and Q samples follow each other alternately (I, Q, I, Q , ...) |

**C Functions**

```
/* Event id */
gecko_evt_cte_receiver_dtm_iq_report_id

/* Event structure */
struct gecko_msg_cte_receiver_dtm_iq_report_evt_t
{
  uint16 status;,
  uint8 channel;,
  int8 rssi;,
  uint8 rssi_antenna_id;,
  uint8 cte_type;,
  uint8 slot_durations;,
  uint16 event_counter;,
  uint8array samples;
};
```

#### 2.2.2.4 evt_cte_receiver_silabs_iq_report

IQ samples report from Silicon Labs CTE packets.

**Table 2.33. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x45 | class | Message class: CTE Receiver |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | status | Status of CTE IQ sampling |
| 6-11 | bd_addr | address | Bluetooth address of the remote device |
| 12 | uint8 | address_type | Advertiser address type. Values:<br>• **0:** Public address<br>• **1:** Random address<br>• **255:** No address provided (anonymous advertising) |
| 13 | uint8 | phy | The PHY on which the packet is received.<br>• **1:** 1M PHY<br>• **2:** 2M PHY |
| 14 | uint8 | channel | The channel on which the CTE packet was received |
| 15 | int8 | rssi | RSSI in the received CTE packet. Unit: dBm |
| 16 | uint8 | rssi_antenna_id | The ID of the antenna on which RSSI was measured |
| 17 | uint8 | cte_type | The CTE type<br>• **0:** AoA CTE response<br>• **1:** AoD CTE response with 1us slots<br>• **2:** AoD CTE response with 2us slots |
| 18 | uint8 | slot_durations | Slot durations<br>• **1:** Switching and sampling slots are 1 us each<br>• **2:** Switching and sampling slots are 2 us each |
| 19-20 | uint16 | packet_counter | The event counter of the periodic advertising train or the connection |
| 21 | uint8array | samples | IQ samples of the received CTE packet. I and Q samples follow each other alternately (I, Q, I, Q , ...) |

**C Functions**

```
/* Event id */
gecko_evt_cte_receiver_silabs_iq_report_id

/* Event structure */
struct gecko_msg_cte_receiver_silabs_iq_report_evt_t
{
  uint16 status;,
  bd_addr address;,
  uint8 address_type;,
  uint8 phy;,
  uint8 channel;,
  int8 rssi;,
  uint8 rssi_antenna_id;,
  uint8 cte_type;,
  uint8 slot_durations;,
```

```
  uint16 packet_counter;,
  uint8array samples;
};
```

## 2.3 CTE Transmitter (cte_transmitter)

Commands and events in this class manage Constant Tone Extension (CTE) transmission.

CTE feature is only supported by specific devices. Commands from this class will return bg_err_not_supported on devices that do not support CTE.

### 2.3.1 cte_transmitter commands

#### 2.3.1.1 cmd_cte_transmitter_clear_dtm_parameters

Clear CTE-related parameters that were previously set for LE transmitter test. Default values will be restored for these parameters.

**Table 2.34. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x05 | method | Message ID |

**Table 2.35. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_cte_transmitter_clear_dtm_parameters_rsp_t *gecko_cmd_cte_transmitter_clear_dtm_parameters();

/* Response id */
gecko_rsp_cte_transmitter_clear_dtm_parameters_id

/* Response structure */
struct gecko_msg_cte_transmitter_clear_dtm_parameters_rsp_t
{
  uint16 result;
};
```

#### 2.3.1.2 cmd_cte_transmitter_disable_connection_cte

Disable CTE responses on a connection.

**Table 2.36. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.37. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_cte_transmitter_disable_connection_cte_rsp_t
*gecko_cmd_cte_transmitter_disable_connection_cte(uint8 connection);

/* Response id */
gecko_rsp_cte_transmitter_disable_connection_cte_id

/* Response structure */
struct gecko_msg_cte_transmitter_disable_connection_cte_rsp_t
{
  uint16 result;
};
```

#### 2.3.1.3 cmd_cte_transmitter_disable_connectionless_cte

Stop the connectionless CTE transmit.

**Table 2.38. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | handle | Periodic advertising handle |

**Table 2.39. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_cte_transmitter_disable_connectionless_cte_rsp_t
*gecko_cmd_cte_transmitter_disable_connectionless_cte(uint8 handle);

/* Response id */
gecko_rsp_cte_transmitter_disable_connectionless_cte_id

/* Response structure */
struct gecko_msg_cte_transmitter_disable_connectionless_cte_rsp_t
{
  uint16 result;
};
```

#### 2.3.1.4 cmd_cte_transmitter_disable_silabs_cte

Disable Silicon Labs CTE transmit.

**Table 2.40. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | handle | Advertising handle |

**Table 2.41. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_cte_transmitter_disable_silabs_cte_rsp_t *gecko_cmd_cte_transmitter_disable_silabs_cte(uint8
handle);

/* Response id */
gecko_rsp_cte_transmitter_disable_silabs_cte_id

/* Response structure */
struct gecko_msg_cte_transmitter_disable_silabs_cte_rsp_t
{
  uint16 result;
};
```

#### 2.3.1.5 cmd_cte_transmitter_enable_connection_cte

Enable different types of CTE responses on a connection. CTE response will be sent once requested by the peer device using the CTE Request procedure.

**Table 2.42. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | cte_types | CTE types. Bitmask of the following:<br>• **Bit 0:** AoA CTE response<br>• **Bit 1:** AoD CTE response with 1 us slots<br>• **Bit 2:** AoD CTE response with 2 us slots |
| 6 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. |

**Table 2.43. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_cte_transmitter_enable_connection_cte_rsp_t
*gecko_cmd_cte_transmitter_enable_connection_cte(uint8     connection,     uint8     cte_types,     uint8
switching_pattern_len, const uint8 *switching_pattern_data);

/* Response id */
gecko_rsp_cte_transmitter_enable_connection_cte_id

/* Response structure */
struct gecko_msg_cte_transmitter_enable_connection_cte_rsp_t
{
  uint16 result;
};
```

#### 2.3.1.6 cmd_cte_transmitter_enable_connectionless_cte

Start connectionless CTE transmit. CTEs will be transmitted in periodic advertisement packets. As a result, a periodic advertising has to be started prior this command.

**Table 2.44. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | handle | Periodic advertising handle |
| 5 | uint8 | cte_length | CTE length in 8 us units.<br>• Range: 0x02 to 0x14<br>• Time Range: 16 us to 160 us |
| 6 | uint8 | cte_type | CTE type<br>• **0:** AoA CTE<br>• **1:** AoD CTE with 1 us slots<br>• **2:** AoD CTE with 2 us slots |
| 7 | uint8 | cte_count | The number of CTEs to be transmitted in each periodic advertising interval |
| 8 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. |

**Table 2.45. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_cte_transmitter_enable_connectionless_cte_rsp_t
*gecko_cmd_cte_transmitter_enable_connectionless_cte(uint8  handle,  uint8  cte_length,  uint8  cte_type,  uint8
cte_count, uint8 switching_pattern_len, const uint8 *switching_pattern_data);

/* Response id */
gecko_rsp_cte_transmitter_enable_connectionless_cte_id

/* Response structure */
struct gecko_msg_cte_transmitter_enable_connectionless_cte_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.3.1.7 cmd_cte_transmitter_enable_silabs_cte

Enable Silicon Labs CTE transmit. CTEs will be transmitted in extended advertisement packets. As a result, extended advertising has to be started prior this command.

**Table 2.46. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | handle | Advertising handle |
| 5 | uint8 | cte_length | CTE length in 8 us units.<br>• Range: 0x02 to 0x14<br>• Time Range: 16 us to 160 us |
| 6 | uint8 | cte_type | CTE type<br>• **0:** AoA CTE<br>• **1:** AoD CTE with 1 us slots<br>• **2:** AoD CTE with 2 us slots |
| 7 | uint8 | cte_count | The number of CTEs to be transmitted in each extended advertising interval. Currently only cte_count = 1 is supported. |
| 8 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. |

**Table 2.47. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_cte_transmitter_enable_silabs_cte_rsp_t  *gecko_cmd_cte_transmitter_enable_silabs_cte(uint8
handle,  uint8  cte_length,  uint8  cte_type,  uint8  cte_count,  uint8  switching_pattern_len,  const  uint8
*switching_pattern_data);

/* Response id */
gecko_rsp_cte_transmitter_enable_silabs_cte_id

/* Response structure */
struct gecko_msg_cte_transmitter_enable_silabs_cte_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.3.1.8 cmd_cte_transmitter_set_dtm_parameters

Set the CTE-related parameters of the LE transmitter test.

**Table 2.48. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | cte_length | Length of the Constant Tone Extension in 8 us units<br>• **0:** No CTE<br>• **0x02 to 0x14:** CTE length<br>Default: 0 (no CTE) |
| 5 | uint8 | cte_type | CTE type<br>• **0:** AoA CTE<br>• **1:** AoD CTE with 1 us slots<br>• **2:** AoD CTE with 2 us slots<br>Default: 0 |
| 6 | uint8array | switching_pattern | Antenna switching pattern. Antennas will be switched in this order with the antenna switch pins during CTE. If the CTE is longer than the switching pattern, the pattern starts over. Default is the empty array. |

**Table 2.49. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x44 | class | Message class: CTE Transmitter |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_cte_transmitter_set_dtm_parameters_rsp_t  *gecko_cmd_cte_transmitter_set_dtm_parameters(uint8
cte_length, uint8 cte_type, uint8 switching_pattern_len, const uint8 *switching_pattern_data);

/* Response id */
gecko_rsp_cte_transmitter_set_dtm_parameters_id

/* Response structure */
struct gecko_msg_cte_transmitter_set_dtm_parameters_rsp_t
{
  uint16 result;
};
```

## 2.4 Device Firmware Upgrade (dfu)

These commands and events are related to controlling firmware updates over the configured host interface and are available only when the device is booted in DFU mode. **DFU process:**

1. Boot device to DFU mode with DFU reset command
2. Wait for DFU boot event
3. Send command Flash Set Address to start the firmware update
4. Upload the firmware with Flash Upload commands until all data is uploaded
5. Send when all data is uploaded
6. Finalize DFU firmware update with Reset command.

DFU mode is using UART baudrate from hardware configuration of firmware. Default baudrate 115200 is used if firmware is missing or firmware content does not match the CRC checksum.

### 2.4.1 dfu commands

#### 2.4.1.1 cmd_dfu_flash_set_address

After re-booting the local device in DFU mode, this command defines the starting address on the flash where the new firmware will be written.

**Table 2.50. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x01 | method | Message ID |
| 4-7 | uint32 | address | The offset in the flash where the new firmware is uploaded to. Always use the value 0x00000000. |

**Table 2.51. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_dfu_flash_set_address_rsp_t *gecko_cmd_dfu_flash_set_address(uint32 address);

/* Response id */
gecko_rsp_dfu_flash_set_address_id

/* Response structure */
struct gecko_msg_dfu_flash_set_address_rsp_t
{
  uint16 result;
};
```

#### 2.4.1.2 cmd_dfu_flash_upload

Upload the whole firmware image file into the Bluetooth device. The passed data length must be a multiple of 4 bytes. Because the BGAPI command payload size is limited, multiple commands need to be issued one after the other until the whole .bin firmware image file is uploaded to the device. After each command, the next address of the flash sector in memory to write to is automatically updated by the bootloader.

**Table 2.52. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x02 | method | Message ID |
| 4 | uint8array | data | An array of data which will be written onto the flash. |

**Table 2.53. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_dfu_flash_upload_rsp_t *gecko_cmd_dfu_flash_upload(uint8 data_len, const uint8 *data_data);

/* Response id */
gecko_rsp_dfu_flash_upload_id

/* Response structure */
struct gecko_msg_dfu_flash_upload_rsp_t
{
  uint16 result;
};
```

#### 2.4.1.3 cmd_dfu_flash_upload_finish

Inform the device that the DFU file is fully uploaded. To return the device back to normal mode, issue the command DFU Reset.

**Table 2.54. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x03 | method | Message ID |

**Table 2.55. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_dfu_flash_upload_finish_rsp_t *gecko_cmd_dfu_flash_upload_finish();

/* Response id */
gecko_rsp_dfu_flash_upload_finish_id

/* Response structure */
struct gecko_msg_dfu_flash_upload_finish_rsp_t
{
  uint16 result;
};
```

#### 2.4.1.4 cmd_dfu_reset

Reset the system. The command does not have a response but it triggers one of the boot events (normal reset or boot to DFU mode) after re-boot.

**Table 2.56. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | dfu | Boot mode:<br>• **0:** Normal reset<br>• **1:** Boot to UART DFU mode<br>• **2:** Boot to OTA DFU mode |

**BGLIB C API**

```
/* Function */
void *gecko_cmd_dfu_reset(uint8 dfu);

/* Command does not have a response */
```

**Table 2.57. Events Generated**

| Event | Description |
|---|---|
| system_boot | Sent after the device has booted in normal mode |
| dfu_boot | Sent after the device has booted in UART DFU mode |

### 2.4.2 dfu events

#### 2.4.2.1 evt_dfu_boot

This event indicates that the device booted in DFU mode and is now ready to receive commands related to device firmware upgrade (DFU).

**Table 2.58. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | version | The version of the bootloader |

**C Functions**

```
/* Event id */
gecko_evt_dfu_boot_id

/* Event structure */
struct gecko_msg_dfu_boot_evt_t
{
  uint32 version;
};
```

#### 2.4.2.2 evt_dfu_boot_failure

This event indicates that an error, which prevents the device from booting, has occurred in bootloader.

**Table 2.59. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x00 | class | Message class: Device Firmware Upgrade |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | reason | The reason for boot failure. See Error codes |

**C Functions**

```
/* Event id */
gecko_evt_dfu_boot_failure_id

/* Event structure */
struct gecko_msg_dfu_boot_failure_evt_t
{
  uint16 reason;
};
```

## 2.5 Persistent Store (flash)

Persistent Store (PS) commands manage user data in PS keys in the flash memory of the Bluetooth device. User data stored within the flash memory is persistent across reset and power cycling of the device. The persistent store size is 2048 bytes. Because Bluetooth bondings are also stored in this area, the space available for user data additionally depends on the number of bondings the device has at the time. The size of a Bluetooth bonding is around 150 bytes.

The maximum user data size associated to a PS key is 56 bytes.

### 2.5.1 flash commands

#### 2.5.1.1 cmd_flash_ps_erase

Delete a single PS key and its value from the persistent store.

**Table 2.60. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | key | PS key to delete |

**Table 2.61. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_flash_ps_erase_rsp_t *gecko_cmd_flash_ps_erase(uint16 key);

/* Response id */
gecko_rsp_flash_ps_erase_id

/* Response structure */
struct gecko_msg_flash_ps_erase_rsp_t
{
  uint16 result;
};
```

#### 2.5.1.2 cmd_flash_ps_erase_all

Delete all PS keys and their corresponding values.

**Table 2.62. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x01 | method | Message ID |

**Table 2.63. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_flash_ps_erase_all_rsp_t *gecko_cmd_flash_ps_erase_all();

/* Response id */
gecko_rsp_flash_ps_erase_all_id

/* Response structure */
struct gecko_msg_flash_ps_erase_all_rsp_t
{
  uint16 result;
};
```

#### 2.5.1.3 cmd_flash_ps_load

Retrieve the value of the specified PS key.

**Table 2.64. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | key | PS key of the value to be retrieved |

**Table 2.65. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | value | The returned value of the specified PS key |

**BGLIB C API**

```
/* Function */
struct gecko_msg_flash_ps_load_rsp_t *gecko_cmd_flash_ps_load(uint16 key);

/* Response id */
gecko_rsp_flash_ps_load_id

/* Response structure */
struct gecko_msg_flash_ps_load_rsp_t
{
  uint16 result;,
  uint8array value;
};
```

#### 2.5.1.4 cmd_flash_ps_save

Store a value into the specified PS key. Allowed PS keys are in range from 0x4000 to 0x407F. At most, 56 bytes user data can be stored in one PS key. Error code 0x018a (command_too_long) is returned if the value data is more than 56 bytes.

**Table 2.66. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | key | PS key |
| 6 | uint8array | value | Value to store into the specified PS key |

**Table 2.67. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0d | class | Message class: Persistent Store |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_flash_ps_save_rsp_t  *gecko_cmd_flash_ps_save(uint16  key,  uint8  value_len,  const  uint8
*value_data);

/* Response id */
gecko_rsp_flash_ps_save_id

/* Response structure */
struct gecko_msg_flash_ps_save_rsp_t
{
  uint16 result;
};
```

### 2.5.2 flash defines

#### 2.5.2.1 define_flash_ps_keys

Define keys

**Table 2.68. Defines**

| Value | Name | Description |
|---|---|---|
| 44 | FLASH_ps_key_local_bd_addr | If defined override address stored during firmware update |
| 49 | FLASH_ps_key_tx_power | Maximum allowed transmitting power |
| 50 | FLASH_ps_key_ctune | Crystal tuning value override |
| 51 | FLASH_ps_key_application_gsn | Application Global State Number value |
| 53 | FLASH_ps_key_ota_flags | OTA configuration flags |
| 54 | FLASH_ps_key_ota_device_name | Device name to be used for OTA |
| 55 | FLASH_ps_key_device_irk | Identity Resolving Key |
| 56 | FLASH_ps_key_bonding_priority_list | Bonding priority list |
| 57 | FLASH_ps_key_ota_advertisement_packet | OTA advertising packet |
| 58 | FLASH_ps_key_ota_scan_response_packet | OTA scan response packet |
| 59 | FLASH_ps_key_application_ai | Application Advertising Identifier value |
| 60 | FLASH_ps_key_identity_addr_type | Identity address type. 0: public (default), 1: static |
| 61 | FLASH_ps_key_gatt_db_hash | GATT DB hash |
| 62 | FLASH_ps_key_ota_rf_path | RF Path used in OTA |
| 16383 | FLASH_ps_key_bonding_db_config | Bonding database configuration |

## 2.6 Generic Attribute Profile (gatt)

The commands and events in this class are used to browse and manage attributes in a remote GATT server.

### 2.6.1 gatt commands

#### 2.6.1.1 cmd_gatt_discover_characteristics

Discover all characteristics of a GATT service from a remote GATT database. This command generates a unique gatt_characteristic event for every discovered characteristic. Received gatt_procedure_completed event indicates that this GATT procedure was successfully completed or failed with an error.

**Table 2.69. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | service | GATT service handle<br>This value is normally received from the gatt_service event. |

**Table 2.70. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct     gecko_msg_gatt_discover_characteristics_rsp_t     *gecko_cmd_gatt_discover_characteristics(uint8
connection, uint32 service);

/* Response id */
gecko_rsp_gatt_discover_characteristics_id

/* Response structure */
struct gecko_msg_gatt_discover_characteristics_rsp_t
{
  uint16 result;
};
```

**Table 2.71. Events Generated**

| Event | Description |
|---|---|
| gatt_characteristic | Discovered characteristic from remote GATT database. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.2 cmd_gatt_discover_characteristics_by_uuid

Discover all characteristics of a GATT service in a remote GATT database having the specified UUID. This command generates a unique gatt_characteristic event for every discovered characteristic having the specified UUID. Received gatt_procedure_completed event indicates that this GATT procedure was successfully completed or failed with an error.

**Table 2.72. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | service | GATT service handle<br>This value is normally received from the gatt_service event. |
| 9 | uint8array | uuid | Characteristic UUID in little endian format |

**Table 2.73. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_discover_characteristics_by_uuid_rsp_t
*gecko_cmd_gatt_discover_characteristics_by_uuid(uint8 connection, uint32 service, uint8 uuid_len, const uint8
*uuid_data);

/* Response id */
gecko_rsp_gatt_discover_characteristics_by_uuid_id

/* Response structure */
struct gecko_msg_gatt_discover_characteristics_by_uuid_rsp_t
{
  uint16 result;
};
```

**Table 2.74. Events Generated**

| Event | Description |
|---|---|
| gatt_characteristic | Discovered characteristic from remote GATT database. |

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.3 cmd_gatt_discover_descriptors

Discover all descriptors of a GATT characteristic in a remote GATT database. It generates a unique gatt_descriptor event for every discovered descriptor. Received gatt_procedure_completed event indicates that this GATT procedure has successfully completed or failed with an error.

**Table 2.75. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |

**Table 2.76. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_gatt_discover_descriptors_rsp_t *gecko_cmd_gatt_discover_descriptors(uint8 connection, uint16
characteristic);

/* Response id */
gecko_rsp_gatt_discover_descriptors_id

/* Response structure */
struct gecko_msg_gatt_discover_descriptors_rsp_t
{
  uint16 result;
};
```

**Table 2.77. Events Generated**

| Event | Description |
|---|---|
| gatt_descriptor | Discovered descriptor from remote GATT database. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.4 cmd_gatt_discover_primary_services

Discover all primary services of a remote GATT database. This command generates a unique gatt_service event for every discovered primary service. Received gatt_procedure_completed event indicates that this GATT procedure has successfully completed or failed with an error.

**Table 2.78. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.79. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_gatt_discover_primary_services_rsp_t    *gecko_cmd_gatt_discover_primary_services(uint8
connection);

/* Response id */
gecko_rsp_gatt_discover_primary_services_id

/* Response structure */
struct gecko_msg_gatt_discover_primary_services_rsp_t
{
  uint16 result;
};
```

**Table 2.80. Events Generated**

| Event | Description |
|---|---|
| gatt_service | Discovered service from remote GATT database |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.5 cmd_gatt_discover_primary_services_by_uuid

Discover primary services with the specified UUID in a remote GATT database. This command generates unique gatt_service event for every discovered primary service. Received gatt_procedure_completed event indicates that this GATT procedure was successfully completed or failed with an error.

**Table 2.81. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8array | uuid | Service UUID in little endian format |

**Table 2.82. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_discover_primary_services_by_uuid_rsp_t
*gecko_cmd_gatt_discover_primary_services_by_uuid(uint8 connection, uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_gatt_discover_primary_services_by_uuid_id

/* Response structure */
struct gecko_msg_gatt_discover_primary_services_by_uuid_rsp_t
{
  uint16 result;
};
```

**Table 2.83. Events Generated**

| Event | Description |
|---|---|
| gatt_service | Discovered service from remote GATT database. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.6 cmd_gatt_execute_characteristic_value_write

Commit or cancel previously queued writes to a long characteristic of a remote GATT server. Writes are sent to the queue with prepare_characteristic_value_write command. Content, offset, and length of queued values are validated by this procedure. A received gatt_procedure_completed event indicates that all data was written successfully or that an error response was received.

**Table 2.84. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0c | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | flags | gatt_execute_write_flag<br>• **0:** cancel<br>• **1:** commit |

**Table 2.85. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_execute_characteristic_value_write_rsp_t
*gecko_cmd_gatt_execute_characteristic_value_write(uint8 connection, uint8 flags);

/* Response id */
gecko_rsp_gatt_execute_characteristic_value_write_id

/* Response structure */
struct gecko_msg_gatt_execute_characteristic_value_write_rsp_t
{
  uint16 result;
};
```

**Table 2.86. Events Generated**

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.7 cmd_gatt_find_included_services

Find the services that are included by a service in a remote GATT database. This command generates a unique gatt_service event for each included service. The received gatt_procedure_completed event indicates that this GATT procedure was successfully completed or failed with an error.

**Table 2.87. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x10 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | service | GATT service handle<br>This value is normally received from the gatt_service event. |

**Table 2.88. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_find_included_services_rsp_t  *gecko_cmd_gatt_find_included_services(uint8  connection,
uint32 service);

/* Response id */
gecko_rsp_gatt_find_included_services_id

/* Response structure */
struct gecko_msg_gatt_find_included_services_rsp_t
{
  uint16 result;
};
```

**Table 2.89. Events Generated**

| Event | Description |
|---|---|
| gatt_service | Discovered service from remote GATT database. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.8 cmd_gatt_prepare_characteristic_value_reliable_write

Add a characteristic value to the write queue of a remote GATT server and verifies whether the value was correctly received by the server. Received gatt_procedure_completed event indicates that this GATT procedure was successfully completed or failed with an error. Specifically, error code 0x0194 (data_corrupted) will be returned if the value received from the GATT server's response fails to pass the reliable write verification. At most ATT_MTU - 5 amount of data can be sent at one time. Writes are executed or canceled with the execute_characteristic_value_write command. Whether the writes succeed or not is indicated in the response of the execute_characteristic_value_write command.

**Table 2.90. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x13 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7-8 | uint16 | offset | Offset of the characteristic value |
| 9 | uint8array | value | Value to write into the specified characteristic of the remote GATT database |

**Table 2.91. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | sent_len | The length of data sent to the remote GATT server |

**BGLIB C API**

```
/* Function */
struct                    gecko_msg_gatt_prepare_characteristic_value_reliable_write_rsp_t
*gecko_cmd_gatt_prepare_characteristic_value_reliable_write(uint8 connection, uint16 characteristic, uint16
offset, uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_prepare_characteristic_value_reliable_write_id

/* Response structure */
struct gecko_msg_gatt_prepare_characteristic_value_reliable_write_rsp_t
{
  uint16 result;,
```

```
  uint16 sent_len;
};
```

**Table 2.92. Events Generated**

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.9 cmd_gatt_prepare_characteristic_value_write

Add a characteristic value to the write queue of a remote GATT server. It can be used when 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 at one time. Writes are executed or canceled with the execute_characteristic_value_write command. Whether the writes succeed or not is indicated in the response of the execute_characteristic_value_write command.

In all use cases where the amount of data to transfer fits into the BGAPI payload, use the command gatt_write_characteristic_value to write long values because it transparently performs the prepare_write and execute_write commands.

**Table 2.93. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0b | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7-8 | uint16 | offset | Offset of the characteristic value |
| 9 | uint8array | value | Value to write into the specified characteristic of the remote GATT database |

**Table 2.94. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | sent_len | The length of data sent to the remote GATT server |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_prepare_characteristic_value_write_rsp_t
*gecko_cmd_gatt_prepare_characteristic_value_write(uint8  connection,  uint16  characteristic,  uint16  offset,
uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_prepare_characteristic_value_write_id

/* Response structure */
struct gecko_msg_gatt_prepare_characteristic_value_write_rsp_t
{
  uint16 result;,
```

```
  uint16 sent_len;
};
```

**Table 2.95. Events Generated**

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.10 cmd_gatt_read_characteristic_value

Read the value of a characteristic from a remote GATT database. A single gatt_characteristic_value event is generated if the characteristic value fits in one ATT PDU. Otherwise, more than one gatt_characteristic_value event is generated because the firmware will automatically use the Read Long Characteristic Values procedure. A received gatt_procedure_completed event indicates that all data was read successfully or that an error response was received.

Note that the GATT client does not verify if the requested attribute is a characteristic value. Therefore, before calling this command, ensure that the attribute handle is for a characteristic value, for example, by performing characteristic discovery.

**Table 2.96. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |

**Table 2.97. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_gatt_read_characteristic_value_rsp_t    *gecko_cmd_gatt_read_characteristic_value(uint8
connection, uint16 characteristic);

/* Response id */
gecko_rsp_gatt_read_characteristic_value_id

/* Response structure */
struct gecko_msg_gatt_read_characteristic_value_rsp_t
{
  uint16 result;
};
```

**Table 2.98. Events Generated**

| Event | Description |
|---|---|
| gatt_characteristic_value | Contains the data of a characteristic sent by the GATT Server. |

| Event | Description |
| --- | --- |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.11 cmd_gatt_read_characteristic_value_by_uuid

Read characteristic values of a service from a remote GATT database by giving the UUID of the characteristic and the handle of the service containing this characteristic. If multiple characteristic values are received in one ATT PDU, then one gatt_characteristic_value event is generated for each value. If the first characteristic value does not fit in one ATT PDU, the firmware automatically uses the Read Long Characteristic Values procedure and generate more gatt_characteristic_value events until the value has been completely read. A received gatt_procedure_completed event indicates that all data was read successfully or that an error response was received.

**Table 2.99. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | service | GATT service handle<br>This value is normally received from the gatt_service event. |
| 9 | uint8array | uuid | Characteristic UUID in little endian format |

**Table 2.100. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_read_characteristic_value_by_uuid_rsp_t
*gecko_cmd_gatt_read_characteristic_value_by_uuid(uint8  connection,  uint32  service,  uint8  uuid_len,  const
uint8 *uuid_data);

/* Response id */
gecko_rsp_gatt_read_characteristic_value_by_uuid_id

/* Response structure */
struct gecko_msg_gatt_read_characteristic_value_by_uuid_rsp_t
{
  uint16 result;
};
```

**Table 2.101. Events Generated**

| Event | Description |
|---|---|
| gatt_characteristic_value | Contains the data of a characteristic sent by the GATT Server. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.12 cmd_gatt_read_characteristic_value_from_offset

Read a partial characteristic value with a specified offset and maximum length from a remote GATT database. It is equivalent to gatt_read_characteristic_value if both the offset and maximum length parameters are 0. A single gatt_characteristic_value event is generated if the value to read fits in one ATT PDU. Otherwise, more than one gatt_characteristic_value events are generated because the firmware will automatically use the Read Long Characteristic Values procedure. A received gatt_procedure_completed event indicates that all data was read successfully or that an error response was received.

**Table 2.102. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x12 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7-8 | uint16 | offset | Offset of the characteristic value |
| 9-10 | uint16 | maxlen | Maximum bytes to read. If this parameter is 0, all characteristic values starting at a given offset will be read. |

**Table 2.103. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_gatt_read_characteristic_value_from_offset_rsp_t
*gecko_cmd_gatt_read_characteristic_value_from_offset(uint8 connection, uint16 characteristic, uint16 offset,
uint16 maxlen);

/* Response id */
gecko_rsp_gatt_read_characteristic_value_from_offset_id

/* Response structure */
struct gecko_msg_gatt_read_characteristic_value_from_offset_rsp_t
{
  uint16 result;
};
```

**Table 2.104. Events Generated**

| Event | Description |
|---|---|
| gatt_characteristic_value | Contains the data of a characteristic sent by the GATT Server. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.13 cmd_gatt_read_descriptor_value

Read the descriptor value of a characteristic in a remote GATT database. A single gatt_descriptor_value event is generated if the descriptor value fits in one ATT PDU. Otherwise, more than one gatt_descriptor_value events are generated because the firmware automatically uses the Read Long Characteristic Values procedure. A received gatt_procedure_completed event indicates that all data was read successfully or that an error response was received.

**Table 2.105. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0e | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | descriptor | GATT characteristic descriptor handle |

**Table 2.106. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_read_descriptor_value_rsp_t  *gecko_cmd_gatt_read_descriptor_value(uint8  connection,
uint16 descriptor);

/* Response id */
gecko_rsp_gatt_read_descriptor_value_id

/* Response structure */
struct gecko_msg_gatt_read_descriptor_value_rsp_t
{
  uint16 result;
};
```

**Table 2.107. Events Generated**

| Event | Description |
|---|---|
| gatt_descriptor_value | Descriptor value received from the remote GATT server. |
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.14 cmd_gatt_read_multiple_characteristic_values

Read values of multiple characteristics from a remote GATT database at once. The GATT server returns values in one ATT PDU as the response. If the total set of values is greater than (ATT_MTU - 1) bytes in length, only the first (ATT_MTU - 1) bytes are included. A single gatt_characteristic_value event is generated, in which the characteristic is set to 0 and data in the value parameter is a concatenation of characteristic values in the order they were requested. The received gatt_procedure_completed event indicates either that this GATT procedure was successfully completed or failed with an error.

Use this command only for characteristics values that have a known fixed size, except the last one that could have variable length.

When the remote GATT server is from Silicon Labs Bluetooth stack, the server returns ATT Invalid PDU (0x04) if this command only reads one characteristic value. The server returns ATT Application Error 0x80 if this command reads the value of a user-type characteristic.

**Table 2.108. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x11 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8array | characteristic_list | List of uint16 characteristic handles each in little endian format. |

**Table 2.109. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_gatt_read_multiple_characteristic_values_rsp_t
*gecko_cmd_gatt_read_multiple_characteristic_values(uint8  connection,  uint8  characteristic_list_len,  const
uint8 *characteristic_list_data);

/* Response id */
gecko_rsp_gatt_read_multiple_characteristic_values_id

/* Response structure */
struct gecko_msg_gatt_read_multiple_characteristic_values_rsp_t
{
  uint16 result;
};
```

**Table 2.110. Events Generated**

| Event | Description |
|---|---|
| gatt_characteristic_value | A concatenation of characteristic values in the order they were requested |
| gatt_procedure_completed | Procedure was either successfully completed or failed with an error. |

#### 2.6.1.15 cmd_gatt_send_characteristic_confirmation

Send a confirmation to a remote GATT server after receiving a characteristic indication. The gatt_characteristic_value event carries the att_opcode containing handle_value_indication (0x1d), which reveals that an indication has been received and must be confirmed with this command. The confirmation needs to be sent within 30 seconds, otherwise further GATT transactions are not allowed by the remote side.

**Table 2.111. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0d | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.112. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_send_characteristic_confirmation_rsp_t
*gecko_cmd_gatt_send_characteristic_confirmation(uint8 connection);

/* Response id */
gecko_rsp_gatt_send_characteristic_confirmation_id

/* Response structure */
struct gecko_msg_gatt_send_characteristic_confirmation_rsp_t
{
  uint16 result;
};
```

#### 2.6.1.16 cmd_gatt_set_characteristic_notification

Enable or disable the notifications and indications 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. A received gatt_procedure_completed event indicates that this GATT procedure was successfully completed or that it failed with an error.

**Table 2.113. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | flags | Characteristic client configuration flags |

**Table 2.114. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_set_characteristic_notification_rsp_t
*gecko_cmd_gatt_set_characteristic_notification(uint8 connection, uint16 characteristic, uint8 flags);

/* Response id */
gecko_rsp_gatt_set_characteristic_notification_id

/* Response structure */
struct gecko_msg_gatt_set_characteristic_notification_rsp_t
{
  uint16 result;
};
```

**Table 2.115. Events Generated**

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

| Event | Description |
|---|---|
| gatt_characteristic_value | If an indication or notification has been enabled for a characteristic, this event is triggered whenever an indication or notification is sent by the remote GATT server. The triggering conditions of the GATT server are defined by an upper level, for example by a profile. **As a result, it is possible that no values are ever received, or that it may take time, depending on how the server is configured.** |

#### 2.6.1.17 cmd_gatt_set_max_mtu

Set the maximum size of ATT Message Transfer Units (MTU). Functionality is the same as gatt_server_set_max_mtu and this setting applies to both GATT client and server. If the given value is too large according to the maximum BGAPI payload size, the system will select the maximum possible value as the maximum ATT_MTU. If maximum ATT_MTU is larger than 23, the GATT client in the stack will automatically send an MTU exchange request after a Bluetooth connection has been established.

**Table 2.116. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | max_mtu | Maximum size of Message Transfer Units (MTU) allowed<br>• Range: 23 to 250<br>Default: 247 |

**Table 2.117. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | max_mtu | The maximum ATT_MTU selected by the system if this command succeeds |

**BGLIB C API**

```
/* Function */
struct gecko_msg_gatt_set_max_mtu_rsp_t *gecko_cmd_gatt_set_max_mtu(uint16 max_mtu);

/* Response id */
gecko_rsp_gatt_set_max_mtu_id

/* Response structure */
struct gecko_msg_gatt_set_max_mtu_rsp_t
{
  uint16 result;,
  uint16 max_mtu;
};
```

#### 2.6.1.18 cmd_gatt_write_characteristic_value

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. Received gatt_procedure_completed event indicates that all data was written successfully or that an error response was received.

**Table 2.118. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8array | value | Characteristic value |

**Table 2.119. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_gatt_write_characteristic_value_rsp_t   *gecko_cmd_gatt_write_characteristic_value(uint8
connection, uint16 characteristic, uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_write_characteristic_value_id

/* Response structure */
struct gecko_msg_gatt_write_characteristic_value_rsp_t
{
  uint16 result;
};
```

**Table 2.120. Events Generated**

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

#### 2.6.1.19 cmd_gatt_write_characteristic_value_without_response

Write the value of a characteristic in a remote GATT server. It does not generate an 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 the database, no error message will be reported to the local GATT client. Note that this command can't be used to write long values. At most ATT_MTU - 3 amount of data can be sent once.

**Table 2.121. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0a | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8array | value | Characteristic value |

**Table 2.122. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | sent_len | The length of data sent to the remote GATT server |

**BGLIB C API**

```
/* Function */
struct                    gecko_msg_gatt_write_characteristic_value_without_response_rsp_t
*gecko_cmd_gatt_write_characteristic_value_without_response(uint8  connection,  uint16  characteristic,  uint8
value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_write_characteristic_value_without_response_id

/* Response structure */
struct gecko_msg_gatt_write_characteristic_value_without_response_rsp_t
{
  uint16 result;,
  uint16 sent_len;
};
```

#### 2.6.1.20 cmd_gatt_write_descriptor_value

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. Received gatt_procedure_completed event indicates either that all data was written successfully or that an error response was received.

**Table 2.123. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0f | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | descriptor | GATT characteristic descriptor handle |
| 7 | uint8array | value | Descriptor value |

**Table 2.124. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_write_descriptor_value_rsp_t  *gecko_cmd_gatt_write_descriptor_value(uint8  connection,
uint16 descriptor, uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_write_descriptor_value_id

/* Response structure */
struct gecko_msg_gatt_write_descriptor_value_rsp_t
{
  uint16 result;
};
```

**Table 2.125. Events Generated**

| Event | Description |
|---|---|
| gatt_procedure_completed | Procedure was successfully completed or failed with an error. |

### 2.6.2 gatt events

#### 2.6.2.1 evt_gatt_characteristic

Indicates that a GATT characteristic in the remote GATT database was discovered. This event is generated after issuing either the gatt_discover_characteristics or command.

**Table 2.126. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle |
| 7 | uint8 | properties | Characteristic properties |
| 8 | uint8array | uuid | Characteristic UUID in little endian format |

**C Functions**

```
/* Event id */
gecko_evt_gatt_characteristic_id

/* Event structure */
struct gecko_msg_gatt_characteristic_evt_t
{
  uint8 connection;,
  uint16 characteristic;,
  uint8 properties;,
  uint8array uuid;
};
```

#### 2.6.2.2 evt_gatt_characteristic_value

Indicates that the value of one or several characteristics in the remote GATT server was received. It is triggered by several commands: gatt_read_characteristic_value, , , gatt_read_multiple_characteristic_values; and when the remote GATT server sends indications or notifications after enabling notifications with gatt_set_characteristic_notification. The parameter att_opcode indicates which type of GATT transaction triggered this event. In particular, if the att_opcode type is handle_value_indication (0x1d), the application needs to confirm the indication with gatt_send_characteristic_confirmation.

**Table 2.127. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | att_opcode | Attribute opcode, which indicates the GATT transaction used |
| 8-9 | uint16 | offset | Value offset |
| 10 | uint8array | value | Characteristic value |

**C Functions**

```
/* Event id */
gecko_evt_gatt_characteristic_value_id

/* Event structure */
struct gecko_msg_gatt_characteristic_value_evt_t
{
  uint8 connection;,
  uint16 characteristic;,
  uint8 att_opcode;,
  uint16 offset;,
  uint8array value;
};
```

#### 2.6.2.3 evt_gatt_descriptor

Indicates that a GATT characteristic descriptor in the remote GATT database was discovered. It is generated after issuing the gatt_discover_descriptors command.

**Table 2.128. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | descriptor | GATT characteristic descriptor handle |
| 7 | uint8array | uuid | Descriptor UUID in little endian format |

**C Functions**

```
/* Event id */
gecko_evt_gatt_descriptor_id

/* Event structure */
struct gecko_msg_gatt_descriptor_evt_t
{
  uint8 connection;,
  uint16 descriptor;,
  uint8array uuid;
};
```

#### 2.6.2.4 evt_gatt_descriptor_value

Indicates that the value of a descriptor in the remote GATT server was received. This event is generated by the gatt_read_descriptor_value command.

**Table 2.129. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | descriptor | GATT characteristic descriptor handle |
| 7-8 | uint16 | offset | Value offset |
| 9 | uint8array | value | Descriptor value |

**C Functions**

```
/* Event id */
gecko_evt_gatt_descriptor_value_id

/* Event structure */
struct gecko_msg_gatt_descriptor_value_evt_t
{
  uint8 connection;,
  uint16 descriptor;,
  uint16 offset;,
  uint8array value;
};
```

#### 2.6.2.5 evt_gatt_mtu_exchanged

Indicates that an ATT_MTU exchange procedure is completed. The mtu parameter describes new MTU size. MTU size 23 is used before this event is received.

**Table 2.130. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | mtu | Exchanged ATT_MTU |

**C Functions**

```
/* Event id */
gecko_evt_gatt_mtu_exchanged_id

/* Event structure */
struct gecko_msg_gatt_mtu_exchanged_evt_t
{
  uint8 connection;,
  uint16 mtu;
};
```

#### 2.6.2.6 evt_gatt_procedure_completed

Indicates that the current GATT procedure was completed successfully or that it failed with an error. All GATT commands excluding gatt_write_characteristic_value_without_response and gatt_send_characteristic_confirmation will trigger this event. As a result, the application must wait for this event before issuing another GATT command (excluding the two aforementioned exceptions).

**Table 2.131. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**C Functions**

```
/* Event id */
gecko_evt_gatt_procedure_completed_id

/* Event structure */
struct gecko_msg_gatt_procedure_completed_evt_t
{
  uint8 connection;,
  uint16 result;
};
```

#### 2.6.2.7 evt_gatt_service

Indicate that a GATT service in the remote GATT database was discovered. This event is generated after issuing either the gatt_discover_primary_services or command.

**Table 2.132. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | service | GATT service handle |
| 9 | uint8array | uuid | Service UUID in little endian format |

**C Functions**

```
/* Event id */
gecko_evt_gatt_service_id

/* Event structure */
struct gecko_msg_gatt_service_evt_t
{
  uint8 connection;,
  uint32 service;,
  uint8array uuid;
};
```

### 2.6.3 gatt enumerations

#### 2.6.3.1 enum_gatt_att_opcode

These values indicate which attribute request or response has caused the event.

**Table 2.133. Enumerations**

| Value | Name | Description |
|---|---|---|
| 8 | gatt_read_by_type_request | Read by type request |
| 9 | gatt_read_by_type_response | Read by type response |
| 10 | gatt_read_request | Read request |
| 11 | gatt_read_response | Read response |
| 12 | gatt_read_blob_request | Read blob request |
| 13 | gatt_read_blob_response | Read blob response |
| 14 | gatt_read_multiple_request | Read multiple request |
| 15 | gatt_read_multiple_response | Read multiple response |
| 18 | gatt_write_request | Write request |
| 19 | gatt_write_response | Write response |
| 82 | gatt_write_command | Write command |
| 22 | gatt_prepare_write_request | Prepare write request |
| 23 | gatt_prepare_write_response | Prepare write response |
| 24 | gatt_execute_write_request | Execute write request |
| 25 | gatt_execute_write_response | Execute write response |
| 27 | gatt_handle_value_notification | Notification |
| 29 | gatt_handle_value_indication | Indication |

#### 2.6.3.2 enum_gatt_client_config_flag

These values define whether the client is to receive notifications or indications from a remote GATT server.

**Table 2.134. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | gatt_disable | Disable notifications and indications |
| 1 | gatt_notification | Notification |
| 2 | gatt_indication | Indication |

#### 2.6.3.3 enum_gatt_execute_write_flag

These values define whether the GATT server is to cancel all queued writes or commit all queued writes to a remote database.

**Table 2.135. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | gatt_cancel | Cancel all queued writes |
| 1 | gatt_commit | Commit all queued writes |

## 2.7 Generic Attribute Profile Server (gatt_server)

These commands and events are used by the local GATT server to manage the local GATT database.

### 2.7.1 gatt_server commands

#### 2.7.1.1 cmd_gatt_server_disable_capabilities

Disable the given capabilities in the local GATT database. See gatt_server_set_capabilities for more formation.

**Table 2.136. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0d | method | Message ID |
| 4-7 | uint32 | caps | Capabilities to disable |

**Table 2.137. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_server_disable_capabilities_rsp_t  *gecko_cmd_gatt_server_disable_capabilities(uint32
caps);

/* Response id */
gecko_rsp_gatt_server_disable_capabilities_id

/* Response structure */
struct gecko_msg_gatt_server_disable_capabilities_rsp_t
{
  uint16 result;
};
```

#### 2.7.1.2 cmd_gatt_server_enable_capabilities

Enable additional capabilities in the local GATT database. Already enabled capabilities keep unchanged after this command. See gatt_server_set_capabilities for more formation.

**Table 2.138. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0c | method | Message ID |
| 4-7 | uint32 | caps | Capabilities to enable |

**Table 2.139. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_gatt_server_enable_capabilities_rsp_t *gecko_cmd_gatt_server_enable_capabilities(uint32 caps);

/* Response id */
gecko_rsp_gatt_server_enable_capabilities_id

/* Response structure */
struct gecko_msg_gatt_server_enable_capabilities_rsp_t
{
  uint16 result;
};
```

#### 2.7.1.3 cmd_gatt_server_find_attribute

Find attributes of a certain type from a local GATT database. The type is usually given as a 16-bit or 128-bit UUID in little endian format.

**Table 2.140. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | start | Search start handle |
| 6 | uint8array | type | The attribute type UUID |

**Table 2.141. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | attribute | Attribute handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_server_find_attribute_rsp_t  *gecko_cmd_gatt_server_find_attribute(uint16  start,  uint8
type_len, const uint8 *type_data);

/* Response id */
gecko_rsp_gatt_server_find_attribute_id

/* Response structure */
struct gecko_msg_gatt_server_find_attribute_rsp_t
{
  uint16 result;,
  uint16 attribute;
};
```

#### 2.7.1.4 cmd_gatt_server_get_enabled_capabilities

Get capabilities currently enabled in the local GATT database.

**Table 2.142. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0e | method | Message ID |

**Table 2.143. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | caps | Enabled capabilities |

**BGLIB C API**

```
/* Function */
struct gecko_msg_gatt_server_get_enabled_capabilities_rsp_t *gecko_cmd_gatt_server_get_enabled_capabilities();

/* Response id */
gecko_rsp_gatt_server_get_enabled_capabilities_id

/* Response structure */
struct gecko_msg_gatt_server_get_enabled_capabilities_rsp_t
{
  uint16 result;,
  uint32 caps;
};
```

#### 2.7.1.5 cmd_gatt_server_get_mtu

Get the size of ATT Message Transfer Units (MTU) for a connection.

**Table 2.144. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0b | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.145. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | mtu | The maximum ATT_MTU used by the connection |

**BGLIB C API**

```
/* Function */
struct gecko_msg_gatt_server_get_mtu_rsp_t *gecko_cmd_gatt_server_get_mtu(uint8 connection);

/* Response id */
gecko_rsp_gatt_server_get_mtu_id

/* Response structure */
struct gecko_msg_gatt_server_get_mtu_rsp_t
{
  uint16 result;,
  uint16 mtu;
};
```

#### 2.7.1.6 cmd_gatt_server_read_attribute_type

Read the type of an attribute from a local GATT database. The type is a UUID, usually 16 or 128 bits long in little endian format.

**Table 2.146. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | attribute | Attribute handle |

**Table 2.147. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | type | The attribute type UUID |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_gatt_server_read_attribute_type_rsp_t   *gecko_cmd_gatt_server_read_attribute_type(uint16
attribute);

/* Response id */
gecko_rsp_gatt_server_read_attribute_type_id

/* Response structure */
struct gecko_msg_gatt_server_read_attribute_type_rsp_t
{
  uint16 result;,
  uint8array type;
};
```

#### 2.7.1.7 cmd_gatt_server_read_attribute_value

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

**Table 2.148. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | attribute | Attribute handle |
| 6-7 | uint16 | offset | Value offset |

**Table 2.149. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | value | The attribute value |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_server_read_attribute_value_rsp_t  *gecko_cmd_gatt_server_read_attribute_value(uint16
attribute, uint16 offset);

/* Response id */
gecko_rsp_gatt_server_read_attribute_value_id

/* Response structure */
struct gecko_msg_gatt_server_read_attribute_value_rsp_t
{
  uint16 result;,
  uint8array value;
};
```

#### 2.7.1.8 cmd_gatt_server_send_characteristic_notification

Send notifications or indications to one or more remote GATT clients. At most ATT_MTU - 3 amount of data can be sent one time.

A notification or indication is sent only if the client has enabled it by setting the corresponding flag to the Client Characteristic Configuration descriptor. If the Client Characteristic Configuration descriptor supports both notifications and indications, the stack will always send a notification even when the client has enabled both.

A new indication to a GATT client can't be sent until an outstanding indication procedure with the same client has completed. The procedure is completed when a confirmation from the client is received. The confirmation is indicated by gatt_server_characteristic_status event.

Error bg_err_wrong_state is returned if the characteristic does not have the notification property, or if the client has not enabled the notification. The same applies to the indication property, and in addition, bg_err_wrong_state is returned if an indication procedure with the same client is outstanding. Always check the response for this command for errors before trying to send more data.

**Table 2.150. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | connection | A handle of the connection over which the notification or indication is sent. Values:<br>• **0xff:** Sends notification or indication to all connected devices.<br>• **Other:** Connection handle |
| 5-6 | uint16 | characteristic | Characteristic handle |
| 7 | uint8array | value | Value to be notified or indicated |

**Table 2.151. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | sent_len | The length of data sent if only one connected device is the receiver; otherwise an unused value. |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_gatt_server_send_characteristic_notification_rsp_t
*gecko_cmd_gatt_server_send_characteristic_notification(uint8   connection,   uint16   characteristic,   uint8
value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_server_send_characteristic_notification_id
```

```
/* Response structure */
struct gecko_msg_gatt_server_send_characteristic_notification_rsp_t
{
  uint16 result;,
  uint16 sent_len;
};
```

#### 2.7.1.9 cmd_gatt_server_send_user_read_response

Send a response to a user_read_request event. The response needs to be sent within 30 seconds, otherwise no more GATT transactions are allowed by the remote side. If attr_errorcode is set to 0, the characteristic value is sent to the remote GATT client in the standard way. Other attr_errorcode values will cause the local GATT server to send an attribute protocol error response instead of the actual data. At most ATT_MTU - 1 amount of data can be sent at one time. The client will continue reading by sending new read request with an increased offset value if it receives ATT_MTU - 1 amount of data.

**Table 2.152. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | att_errorcode | Attribute protocol error code<br>• **0:** No error<br>• **Non-zero:** See Bluetooth specification, Host volume, Attribute Protocol, Error Codes table. |
| 8 | uint8array | value | Characteristic value to send to the GATT client. Ignored if att_errorcode is not 0. |

**Table 2.153. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | sent_len | The length of data sent to the remote GATT client |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_server_send_user_read_response_rsp_t
*gecko_cmd_gatt_server_send_user_read_response(uint8 connection, uint16 characteristic, uint8 att_errorcode,
uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_server_send_user_read_response_id

/* Response structure */
struct gecko_msg_gatt_server_send_user_read_response_rsp_t
{
  uint16 result;,
```

```
  uint16 sent_len;
};
```

#### 2.7.1.10 cmd_gatt_server_send_user_write_response

Send a response to a gatt_server_user_write_request event when parameter att_opcode in the event is Write Request (see att_opcode). The response needs to be sent within 30 seconds, otherwise no more GATT transactions are allowed by the remote side. If attr_errorcode is set to 0, the ATT protocol's write response is sent to indicate to the remote GATT client that the write operation was processed successfully. Other values will cause the local GATT server to send an ATT protocol error response.

**Table 2.154. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | att_errorcode | Attribute protocol error code<br>• **0:** No error<br>• **Non-zero:** See Bluetooth specification, Host volume, Attribute Protocol, Error Codes table. |

**Table 2.155. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_gatt_server_send_user_write_response_rsp_t
*gecko_cmd_gatt_server_send_user_write_response(uint8 connection, uint16 characteristic, uint8 att_errorcode);

/* Response id */
gecko_rsp_gatt_server_send_user_write_response_id

/* Response structure */
struct gecko_msg_gatt_server_send_user_write_response_rsp_t
{
  uint16 result;
};
```

#### 2.7.1.11 cmd_gatt_server_set_capabilities

Reset capabilities that should be enabled by the GATT database. A service is visible to remote GATT clients if at least one of its capabilities is enabled. The same applies to a characteristic and its attributes. Capability identifiers and their corresponding bit flag values can be found in the auto-generated database header file. See UG118: Blue Gecko Bluetooth Profile Toolkit Developer's Guide for how to declare capabilities in the GATT database.

Changing the capabilities of a database effectively causes a database change (attributes being added or removed) from a remote GATT client point of view. If the database has a Generic Attribute service and Service Changed characteristic, the stack will monitor the local database change status and manage service changed indications for a GATT client that has enabled the indication configuration of the Service Changed characteristic.

**Table 2.156. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x08 | method | Message ID |
| 4-7 | uint32 | caps | Bit flags of capabilities to reset. Value 0 sets the default database capabilities. |
| 8-11 | uint32 | reserved | Use the value 0 on this reserved field. Do not use none-zero values because they are reserved for future use. |

**Table 2.157. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_server_set_capabilities_rsp_t  *gecko_cmd_gatt_server_set_capabilities(uint32  caps,
uint32 reserved);

/* Response id */
gecko_rsp_gatt_server_set_capabilities_id

/* Response structure */
struct gecko_msg_gatt_server_set_capabilities_rsp_t
{
  uint16 result;
};
```

#### 2.7.1.12 cmd_gatt_server_set_max_mtu

Set the maximum size of ATT Message Transfer Units (MTU). The functionality is the same as gatt_set_max_mtu and this setting applies to both GATT client and server. If the given value is too large according to the maximum BGAPI payload size, the system will select the maximum possible value as the maximum ATT_MTU. If the maximum ATT_MTU is larger than 23, the GATT client in the stack will automatically send an MTU exchange request after a Bluetooth connection was established.

**Table 2.158. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | max_mtu | Maximum size of Message Transfer Units (MTU) allowed<br>• Range: 23 to 250<br>Default: 247 |

**Table 2.159. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | max_mtu | The maximum ATT_MTU selected by the system if this command succeeded |

**BGLIB C API**

```
/* Function */
struct gecko_msg_gatt_server_set_max_mtu_rsp_t *gecko_cmd_gatt_server_set_max_mtu(uint16 max_mtu);

/* Response id */
gecko_rsp_gatt_server_set_max_mtu_id

/* Response structure */
struct gecko_msg_gatt_server_set_max_mtu_rsp_t
{
  uint16 result;,
  uint16 max_mtu;
};
```

#### 2.7.1.13 cmd_gatt_server_write_attribute_value

Write the value of an attribute in the local GATT database. Writing the value of a characteristic of the local GATT database will not trigger notifications or indications to the remote GATT client if the characteristic has a property to indicate or notify and the client has enabled notification or indication. Notifications and indications are sent to the remote GATT client using gatt_server_send_characteristic_notification command.

**Table 2.160. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | attribute | Attribute handle |
| 6-7 | uint16 | offset | Value offset |
| 8 | uint8array | value | Value |

**Table 2.161. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_gatt_server_write_attribute_value_rsp_t  *gecko_cmd_gatt_server_write_attribute_value(uint16
attribute, uint16 offset, uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_gatt_server_write_attribute_value_id

/* Response structure */
struct gecko_msg_gatt_server_write_attribute_value_rsp_t
{
  uint16 result;
};
```

### 2.7.2 gatt_server events

#### 2.7.2.1 evt_gatt_server_attribute_value

Indicates that the value of an attribute in the local GATT database was changed by a remote GATT client. The parameter att_opcode describes which GATT procedure was used to change the value.

**Table 2.162. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | attribute | Attribute Handle |
| 7 | uint8 | att_opcode | Attribute opcode that informs the procedure from which the value was received. |
| 8-9 | uint16 | offset | Value offset |
| 10 | uint8array | value | Value |

**C Functions**

```
/* Event id */
gecko_evt_gatt_server_attribute_value_id

/* Event structure */
struct gecko_msg_gatt_server_attribute_value_evt_t
{
  uint8 connection;,
  uint16 attribute;,
  uint8 att_opcode;,
  uint16 offset;,
  uint8array value;
};
```

#### 2.7.2.2 evt_gatt_server_characteristic_status

Indicates either that a local Client Characteristic Configuration descriptor was changed by the remote GATT client, or that a confirmation from the remote GATT client was received upon a successful reception of the indication. Confirmation by the remote GATT client should be received within 30 seconds after an indication was sent with the gatt_server_send_characteristic_notification command, otherwise further GATT transactions over this connection are not allowed by the stack.

**Table 2.163. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br><br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | status_flags | Describes whether Client Characteristic Configuration was changed or if a confirmation was received. |
| 8-9 | uint16 | client_config_flags | This field carries the new value of the Client Characteristic Configuration. If the status_flags is 0x2 (confirmation received), the value of this field can be ignored. |

**C Functions**

```
/* Event id */
gecko_evt_gatt_server_characteristic_status_id

/* Event structure */
struct gecko_msg_gatt_server_characteristic_status_evt_t
{
  uint8 connection;,
  uint16 characteristic;,
  uint8 status_flags;,
  uint16 client_config_flags;
};
```

#### 2.7.2.3 evt_gatt_server_execute_write_completed

Execute write completed event indicates that the execute write command from a remote GATT client has completed with the given result.

**Table 2.164. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | result | Execute write result |

**C Functions**

```
/* Event id */
gecko_evt_gatt_server_execute_write_completed_id

/* Event structure */
struct gecko_msg_gatt_server_execute_write_completed_evt_t
{
  uint8 connection;,
  uint16 result;
};
```

#### 2.7.2.4 evt_gatt_server_user_read_request

Indicates that a remote GATT client is attempting to read a value of an attribute from the local GATT database, where the attribute was defined in the GATT database XML file to have the type="user". The parameter att_opcode informs which GATT procedure was used to read the value. The application needs to respond to this request by using the gatt_server_send_user_read_response command within 30 seconds, otherwise further GATT transactions are not allowed by the remote side.

**Table 2.165. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br><br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | att_opcode | Attribute opcode that informs the procedure from which the value was received. |
| 8-9 | uint16 | offset | Value offset |

**C Functions**

```
/* Event id */
gecko_evt_gatt_server_user_read_request_id

/* Event structure */
struct gecko_msg_gatt_server_user_read_request_evt_t
{
  uint8 connection;,
  uint16 characteristic;,
  uint8 att_opcode;,
  uint16 offset;
};
```

#### 2.7.2.5 evt_gatt_server_user_write_request

Indicates that a remote GATT client is attempting to write a value of an attribute into the local GATT database, where the attribute was defined in the GATT database XML file to have the type="user". The parameter att_opcode informs which attribute procedure was used to write the value. If the att_opcode is Write Request (see att_opcode), the application needs to respond to this request by using the gatt_server_send_user_write_response command within 30 seconds, otherwise further GATT transactions are not allowed by the remote side. If the value of att_opcode is Execute Write Request, it indicates that this is a queued prepare write request received earlier and now the GATT server is processing the execute write. The event gatt_server_execute_write_completed will be emitted after all queued requests have been processed.

**Table 2.166. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | characteristic | GATT characteristic handle<br><br>This value is normally received from the gatt_characteristic event. |
| 7 | uint8 | att_opcode | Attribute opcode that informs the procedure from which the value was received. |
| 8-9 | uint16 | offset | Value offset |
| 10 | uint8array | value | Value |

**C Functions**

```
/* Event id */
gecko_evt_gatt_server_user_write_request_id

/* Event structure */
struct gecko_msg_gatt_server_user_write_request_evt_t
{
  uint8 connection;,
  uint16 characteristic;,
  uint8 att_opcode;,
  uint16 offset;,
  uint8array value;
};
```

### 2.7.3 gatt_server enumerations

#### 2.7.3.1 enum_gatt_server_characteristic_status_flag

These values describe whether the characteristic client configuration was changed or whether a characteristic confirmation was received.

**Table 2.167. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | gatt_server_client_config | Characteristic client configuration has been changed. |
| 2 | gatt_server_confirmation | Characteristic confirmation has been received. |

## 2.8 Hardware (hardware)

The commands and events in this class access and configure the system hardware and peripherals.

### 2.8.1 hardware commands

#### 2.8.1.1 (deprecated) cmd_hardware_get_time

Deprecated. Use Sleep Timer component (sl_sleeptimer.h) for the same functionality. Call sl_sleeptimer_get_tick_count64 to get current tick count. Sleep Timer provides APIs for conversions between ticks and milliseconds.

Get elapsed time since last reset.

**Table 2.168. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0b | method | Message ID |

**Table 2.169. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0b | method | Message ID |
| 4-7 | uint32 | seconds | Seconds since last reset |
| 8-9 | uint16 | ticks | Subsecond ticks of hardware clock, range 0-32767 |

**BGLIB C API**

```
/* Function */
struct gecko_msg_hardware_get_time_rsp_t *gecko_cmd_hardware_get_time();

/* Response id */
gecko_rsp_hardware_get_time_id

/* Response structure */
struct gecko_msg_hardware_get_time_rsp_t
{
  uint32 seconds;,
  uint16 ticks;
};
```

#### 2.8.1.2 cmd_hardware_set_lazy_soft_timer

Start a software timer with slack. The slack parameter allows the stack to optimize wakeups and save power. The timer event is triggered between time and time + slack. See also description of hardware_set_soft_timer command.

**Table 2.170. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0c | method | Message ID |
| 4-7 | uint32 | time | Interval between how often to send events in hardware clock ticks (1 second is equal to 32768 ticks).<br>The smallest interval value supported is 328, which is around 10 milliseconds. Any parameters between 0 and 328 will be rounded up to 328. The maximum value is 2147483647, which corresponds to about 18.2 hours.<br>If time is 0, removes the scheduled timer with the same handle. |
| 8-11 | uint32 | slack | Slack time in hardware clock ticks |
| 12 | uint8 | handle | Timer handle to use, which is returned in timeout event |
| 13 | uint8 | single_shot | Timer mode. Values:<br>• **0:** false (timer is repeating)<br>• **1:** true (timer runs only once) |

**Table 2.171. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_hardware_set_lazy_soft_timer_rsp_t  *gecko_cmd_hardware_set_lazy_soft_timer(uint32  time,
uint32 slack, uint8 handle, uint8 single_shot);

/* Response id */
gecko_rsp_hardware_set_lazy_soft_timer_id

/* Response structure */
struct gecko_msg_hardware_set_lazy_soft_timer_rsp_t
{
  uint16 result;
};
```

**Table 2.172. Events Generated**

| Event | Description |
|---|---|
| hardware_soft_timer | Sent after this timer has lapsed. |

#### 2.8.1.3 cmd_hardware_set_soft_timer

Start a software timer. Multiple concurrent timers can be running simultaneously. 256 unique timer handles (IDs) are available. The maximum number of concurrent timers is configurable at device initialization. Up to 16 concurrent timers can be configured. The default configuration is 4. As the RAM for storing timer data is pre-allocated at initialization, an application should not configure the amount more than it needs for minimizing RAM usage.

**Table 2.173. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | time | Frequency interval of events, which indicates how often to send events in hardware clock ticks (1 second is equal to 32768 ticks).<br><br>The smallest interval value supported is 328, which is around 10 milliseconds. Any parameters between 0 and 328 will be rounded up to 328. The maximum value is 2147483647, which corresponds to about 18.2 hours.<br>If time is 0, removes the scheduled timer with the same handle. |
| 8 | uint8 | handle | Timer handle to use, which is returned in timeout event |
| 9 | uint8 | single_shot | Timer mode. Values:<br>• **0:** false (timer is repeating)<br>• **1:** true (timer runs only once) |

**Table 2.174. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_hardware_set_soft_timer_rsp_t *gecko_cmd_hardware_set_soft_timer(uint32 time, uint8 handle,
uint8 single_shot);

/* Response id */
gecko_rsp_hardware_set_soft_timer_id

/* Response structure */
struct gecko_msg_hardware_set_soft_timer_rsp_t
{
  uint16 result;
};
```

**Table 2.175. Events Generated**

| Event | Description |
|---|---|
| hardware_soft_timer | Sent after this timer has lapsed. |

### 2.8.2 hardware events

#### 2.8.2.1 evt_hardware_soft_timer

Indicates that a soft timer has lapsed.

**Table 2.176. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | handle | Timer Handle |

**C Functions**

```
/* Event id */
gecko_evt_hardware_soft_timer_id

/* Event structure */
struct gecko_msg_hardware_soft_timer_evt_t
{
  uint8 handle;
};
```

## 2.9 Connection Management (le_connection)

The commands and events in this class are related to managing connection establishment, parameter setting, and disconnection procedures.

### 2.9.1 le_connection commands

#### 2.9.1.1 cmd_le_connection_close

Close a Bluetooth connection or cancel an ongoing connection establishment process. The parameter is a connection handle which is reported in le_connection_opened event or le_gap_connect response.

**Table 2.177. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Handle of the connection to be closed |

**Table 2.178. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_connection_close_rsp_t *gecko_cmd_le_connection_close(uint8 connection);

/* Response id */
gecko_rsp_le_connection_close_id

/* Response structure */
struct gecko_msg_le_connection_close_rsp_t
{
  uint16 result;
};
```

**Table 2.179. Events Generated**

| Event | Description |
|---|---|
| le_connection_closed | Indicates that a connection was closed. |

#### 2.9.1.2 cmd_le_connection_disable_slave_latency

Temporarily enable or disable slave latency. Used only when Bluetooth device is acting as slave. When slave latency is disabled, the slave latency connection parameter is not set to 0 but the device will wake up on every connection interval to receive and send packets.

**Table 2.180. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection Handle |
| 5 | uint8 | disable | 0 enable, 1 disable slave latency. Default: 0 |

**Table 2.181. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_connection_disable_slave_latency_rsp_t
*gecko_cmd_le_connection_disable_slave_latency(uint8 connection, uint8 disable);

/* Response id */
gecko_rsp_le_connection_disable_slave_latency_id

/* Response structure */
struct gecko_msg_le_connection_disable_slave_latency_rsp_t
{
  uint16 result;
};
```

#### 2.9.1.3 cmd_le_connection_get_rssi

Get the latest RSSI value of a Bluetooth connection. The RSSI value will be reported in a le_connection_rssi event.

**Table 2.182. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.183. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_connection_get_rssi_rsp_t *gecko_cmd_le_connection_get_rssi(uint8 connection);

/* Response id */
gecko_rsp_le_connection_get_rssi_id

/* Response structure */
struct gecko_msg_le_connection_get_rssi_rsp_t
{
  uint16 result;
};
```

**Table 2.184. Events Generated**

| Event | Description |
|---|---|
| le_connection_rssi | Triggered when this command has completed. |

#### 2.9.1.4 cmd_le_connection_read_channel_map

Read channel map for a specified connection.

**Table 2.185. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | connection | Connection Handle |

**Table 2.186. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | channel_map | This parameter is 5 bytes and contains 37 1-bit fields. The nth field (in the range 0 to 36) contains the value for the link layer channel index n.<br>• **0:** Channel n is unused.<br>• **1:** Channel n is used.<br>The most significant bits are reserved for future use. |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_le_connection_read_channel_map_rsp_t    *gecko_cmd_le_connection_read_channel_map(uint8
connection);

/* Response id */
gecko_rsp_le_connection_read_channel_map_id

/* Response structure */
struct gecko_msg_le_connection_read_channel_map_rsp_t
{
  uint16 result;,
  uint8array channel_map;
};
```

#### 2.9.1.5 (deprecated) cmd_le_connection_set_parameters

**Deprecated** and replaced by le_connection_set_timing_parameters command.

Request a change in the connection parameters of a Bluetooth connection.

**Table 2.187. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | connection | Connection Handle |
| 5-6 | uint16 | min_interval | Minimum value for the connection event interval. This must be set be less than or equal to max_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s |
| 7-8 | uint16 | max_interval | Maximum value for the connection event interval. This must be set greater than or equal to min_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s |
| 9-10 | uint16 | latency | Slave latency, which defines how many connection intervals the slave can skip if it has no data to send<br>• Range: 0x0000 to 0x01f4<br>Use 0x0000 for default value |
| 11-12 | uint16 | timeout | Supervision timeout, which defines the time that the connection is maintained although the devices can't communicate at the currently configured connection intervals.<br>• Range: 0x000a to 0x0c80<br>• Time = Value x 10 ms<br>• Time Range: 100 ms to 32 s<br>• The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds<br>Set the supervision timeout at a value which allows communication attempts over at least a few connection intervals. |

**Table 2.188. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x00 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_connection_set_parameters_rsp_t *gecko_cmd_le_connection_set_parameters(uint8 connection,
uint16 min_interval, uint16 max_interval, uint16 latency, uint16 timeout);

/* Response id */
gecko_rsp_le_connection_set_parameters_id

/* Response structure */
struct gecko_msg_le_connection_set_parameters_rsp_t
{
  uint16 result;
};
```

**Table 2.189. Events Generated**

| Event | Description |
|---|---|
| le_connection_parameters | Triggered after new connection parameters are applied on the connection. |

#### 2.9.1.6 (deprecated) cmd_le_connection_set_phy

**Deprecated** and replaced by le_connection_set_preferred_phy command.

Set preferred PHYs for a connection. Preferred PHYs are connection-specific. Event le_connection_phy_status is received when PHY update procedure is completed. Non-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.

**Table 2.190. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | |
| 5 | uint8 | phy | Preferred PHYs for connection. This parameter is a bitfield and multiple PHYs can be preferred by setting multiple bits.<br>• **0x01:** 1M PHY<br>• **0x02:** 2M PHY<br>• **0x04:** 125k Coded PHY (S=8)<br>• **0x08:** 500k Coded PHY (S=2) |

**Table 2.191. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_connection_set_phy_rsp_t *gecko_cmd_le_connection_set_phy(uint8 connection, uint8 phy);

/* Response id */
gecko_rsp_le_connection_set_phy_id

/* Response structure */
struct gecko_msg_le_connection_set_phy_rsp_t
{
  uint16 result;
};
```

**Table 2.192. Events Generated**

| Event | Description |
|---|---|
| le_connection_phy_status | Indicates that PHY update procedure is completed. |

#### 2.9.1.7 cmd_le_connection_set_preferred_phy

Sets preferred and accepted PHYs for the given connection. Event le_connection_phy_status is received when PHY update procedure is completed. Non-preferred PHY can also be set if remote device does not accept any of the preferred PHYs.

The parameter accepted_phy is used for specifying the PHYs that the stack can accept in a remote initiated PHY update request. A PHY update will not occur if none of the accepted PHYs presents in the request.

**NOTE:** 2M and Coded PHYs are not supported by all devices.

**Table 2.193. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | preferred_phy | Preferred PHYs. This parameter is a bitfield and multiple PHYs can be set.<br>• **0x01:** 1M PHY<br>• **0x02:** 2M PHY<br>• **0x04:** 125k Coded PHY (S=8)<br>• **0x08:** 500k Coded PHY (S=2)<br>Default: 0xff (no preference) |
| 6 | uint8 | accepted_phy | Accepted PHYs in remotely-initiated PHY update requests. This parameter is a bitfield and multiple PHYs can be set.<br>• **0x01:** 1M PHY<br>• **0x02:** 2M PHY<br>• **0x04:** Coded PHY<br>• **0xff:** Any PHYs<br>Default: 0xff (all PHYs accepted) |

**Table 2.194. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_le_connection_set_preferred_phy_rsp_t   *gecko_cmd_le_connection_set_preferred_phy(uint8
connection, uint8 preferred_phy, uint8 accepted_phy);

/* Response id */
```

```
gecko_rsp_le_connection_set_preferred_phy_id

/* Response structure */
struct gecko_msg_le_connection_set_preferred_phy_rsp_t
{
  uint16 result;
};
```

**Table 2.195. Events Generated**

| Event | Description |
|---|---|
| le_connection_phy_status | Indicates that PHY update procedure is completed. |

#### 2.9.1.8 cmd_le_connection_set_timing_parameters

Request a change in the connection parameters of a Bluetooth connection.

**Table 2.196. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | connection | Connection Handle |
| 5-6 | uint16 | min_interval | Minimum value for the connection event interval. This must be set less than or equal to max_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s |
| 7-8 | uint16 | max_interval | Maximum value for the connection event interval. This must be set greater than or equal to min_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s |
| 9-10 | uint16 | latency | Slave latency, which defines how many connection intervals the slave can skip if it has no data to send<br>• Range: 0x0000 to 0x01f4<br>Use 0x0000 for default value |
| 11-12 | uint16 | timeout | Supervision timeout, which defines the time that the connection is maintained although the devices can't communicate at the currently configured connection intervals.<br>• Range: 0x000a to 0x0c80<br>• Time = Value x 10 ms<br>• Time Range: 100 ms to 32 s<br>• The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds<br>Set the supervision timeout at a value which allows communication attempts over at least a few connection intervals. |
| 13-14 | uint16 | min_ce_length | Minimum value for the connection event length. This must be set less than or equal to max_ce_length.<br>• Time = Value x 0.625 ms<br>• Range: 0x0000 to 0xffff<br>Value is not currently used and is reserved for future. Set to 0. |
| 15-16 | uint16 | max_ce_length | Maximum value for the connection event length. This must be set greater than or equal to min_ce_length.<br>• Time = Value x 0.625 ms<br>• Range: 0x0000 to 0xffff<br>Use 0xffff for no limitation. |

**Table 2.197. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |

| Byte | Type | Name | Description |
|---|---|---|---|
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_connection_set_timing_parameters_rsp_t
*gecko_cmd_le_connection_set_timing_parameters(uint8 connection, uint16 min_interval, uint16 max_interval,
uint16 latency, uint16 timeout, uint16 min_ce_length, uint16 max_ce_length);

/* Response id */
gecko_rsp_le_connection_set_timing_parameters_id

/* Response structure */
struct gecko_msg_le_connection_set_timing_parameters_rsp_t
{
  uint16 result;
};
```

**Table 2.198. Events Generated**

| Event | Description |
|---|---|
| le_connection_parameters | Triggered after new connection parameters are applied on the connection. |

### 2.9.2 le_connection events

#### 2.9.2.1 evt_le_connection_closed

Indicates that a connection was closed.

**Table 2.199. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | reason | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | connection | Handle of the closed connection |

**C Functions**

```
/* Event id */
gecko_evt_le_connection_closed_id

/* Event structure */
struct gecko_msg_le_connection_closed_evt_t
{
  uint16 reason;,
  uint8 connection;
};
```

#### 2.9.2.2 evt_le_connection_opened

Indicates that a new connection was opened. This event does not indicate that the connection was established (i.e., that a data packet was received within 6 connection interval). If the connection does not get established, an le_connection_closed event may immediately follow. This event also reports whether the connected devices are already bonded, and what the role of the Bluetooth device (Slave or Master) is. An open connection can be closed with the le_connection_close command by giving the connection handle obtained from this event.

**Table 2.200. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Remote device address |
| 10 | uint8 | address_type | Remote device address type |
| 11 | uint8 | master | Device role in connection. Values:<br>• **0:** Slave<br>• **1:** Master |
| 12 | uint8 | connection | Handle for new connection |
| 13 | uint8 | bonding | Bonding handle. Values:<br>• **0xff:** No bonding<br>• **Other:** Bonding handle |
| 14 | uint8 | advertiser | The local advertising set that this connection was opened to. Values:<br>• **0xff:** Invalid value or not applicable. Ignore this field<br>• **Other:** The advertising set handle |

**C Functions**

```
/* Event id */
gecko_evt_le_connection_opened_id

/* Event structure */
struct gecko_msg_le_connection_opened_evt_t
{
  bd_addr address;,
  uint8 address_type;,
  uint8 master;,
  uint8 connection;,
  uint8 bonding;,
  uint8 advertiser;
};
```

#### 2.9.2.3 evt_le_connection_parameters

Triggered whenever the connection parameters are changed and at any time a connection is established.

**Table 2.201. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | interval | Connection interval. Time = Value x 1.25 ms |
| 7-8 | uint16 | latency | Slave latency (how many connection intervals the slave can skip) |
| 9-10 | uint16 | timeout | Supervision timeout. Time = Value x 10 ms |
| 11 | uint8 | security_mode | Connection security mode |
| 12-13 | uint16 | txsize | Maximum Data Channel PDU Payload size that the controller can send in an air packet |

**C Functions**

```
/* Event id */
gecko_evt_le_connection_parameters_id

/* Event structure */
struct gecko_msg_le_connection_parameters_evt_t
{
  uint8 connection;,
  uint16 interval;,
  uint16 latency;,
  uint16 timeout;,
  uint8 security_mode;,
  uint16 txsize;
};
```

#### 2.9.2.4 evt_le_connection_phy_status

Indicates that PHY update procedure is completed.

**Table 2.202. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | phy | Current active PHY. See values from le_connection_set_preferred_phy command. |

**C Functions**

```
/* Event id */
gecko_evt_le_connection_phy_status_id

/* Event structure */
struct gecko_msg_le_connection_phy_status_evt_t
{
  uint8 connection;,
  uint8 phy;
};
```

#### 2.9.2.5 evt_le_connection_rssi

Triggered when an le_connection_get_rssi command has completed.

**Table 2.203. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection Management |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | status | Command complete status:<br>• **0x00:** The command succeeded<br>• **0x01-0xFF:** The command failed. See Bluetooth Core specification v5.0 [Vol 2] Part D, Error Codes |
| 6 | int8 | rssi | RSSI in the latest received packet of the connection. Units: dBm. Range: -127 to +20. Ignore this parameter if the command fails. |

**C Functions**

```
/* Event id */
gecko_evt_le_connection_rssi_id

/* Event structure */
struct gecko_msg_le_connection_rssi_evt_t
{
  uint8 connection;,
  uint8 status;,
  int8 rssi;
};
```

### 2.9.3 le_connection enumerations

#### 2.9.3.1 enum_le_connection_security

Indicate the Bluetooth Security Mode.

**Table 2.204. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | le_connection_mode1_level1 | No security |
| 1 | le_connection_mode1_level2 | Unauthenticated pairing with encryption |
| 2 | le_connection_mode1_level3 | Authenticated pairing with encryption |
| 3 | le_connection_mode1_level4 | Authenticated Secure Connections pairing with encryption using a 128-bit strength encryption key |

## 2.10 Generic Access Profile (le_gap)

The commands and events in this class are related to the Generic Access Profile (GAP) in Bluetooth.

### 2.10.1 le_gap commands

#### 2.10.1.1 cmd_le_gap_bt5_set_adv_data

Set user-defined data in advertising packets, scan response packets, or periodic advertising packets. Maximum 31 bytes of data can be set for legacy advertising. Maximum 191 bytes of data can be set for connectable extended advertising. Maximum 253 bytes of data can be set for periodic and non-connectable extended advertising. For setting longer advertising data, use command le_gap_set_long_advertising_data.

If advertising mode is currently enabled, the new advertising data will be used immediately. Advertising mode can be enabled using command le_gap_start_advertising. Periodic advertising mode can be enabled using command le_gap_start_periodic_advertising.

The invalid parameter error will be returned in the following situations:

- Data length is more than 31 bytes but the advertiser can only advertise using legacy advertising PDUs.
- Data is too long to fit into a single advertisement.
- Set data of the advertising data packet when the advertiser is advertising in scannable mode using extended advertising PDUs.
- Set data of the 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 a discoverable mode other than user_data.

**Table 2.205. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0c | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | scan_rsp | This value selects whether data is intended for advertising packets, scan response packets, periodic advertising packets, or advertising packets in OTA. Values are as follows:<br>• **0:** Advertising packets<br>• **1:** Scan response packets<br>• **2:** OTA advertising packets<br>• **4:** OTA scan response packets<br>• **8:** Periodic advertising packets |
| 6 | uint8array | adv_data | Data to be set |

**Table 2.206. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_bt5_set_adv_data_rsp_t  *gecko_cmd_le_gap_bt5_set_adv_data(uint8  handle,  uint8
scan_rsp, uint8 adv_data_len, const uint8 *adv_data_data);

/* Response id */
gecko_rsp_le_gap_bt5_set_adv_data_id

/* Response structure */
struct gecko_msg_le_gap_bt5_set_adv_data_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.2 (deprecated) cmd_le_gap_bt5_set_adv_parameters

**Deprecated** and replaced by le_gap_set_advertise_timing command to set the advertising intervals, le_gap_set_advertise_channel_map command to set the channel map, and le_gap_set_advertise_report_scan_request command to enable and disable scan request notifications.

**Table 2.207. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0b | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5-6 | uint16 | interval_min | Minimum advertising interval. Value in units of 0.625 ms<br>• Range: 0x20 to 0xFFFF<br>• Time range: 20 ms to 40.96 s<br>Default value: 100 ms |
| 7-8 | uint16 | interval_max | Maximum advertising interval. Value in units of 0.625 ms<br>• Range: 0x20 to 0xFFFF<br>• Time range: 20 ms to 40.96 s<br>• Note: interval_max should be bigger than interval_min<br>Default value: 200 ms |
| 9 | uint8 | channel_map | Advertising channel map, which determines which of the three channels will be used for advertising. This value is given as a bitmask. Values:<br>• **1:** Advertise on CH37<br>• **2:** Advertise on CH38<br>• **3:** Advertise on CH37 and CH38<br>• **4:** Advertise on CH39<br>• **5:** Advertise on CH37 and CH39<br>• **6:** Advertise on CH38 and CH39<br>• **7:** Advertise on all channels<br>Recommended value: 7<br>Default value: 7 |
| 10 | uint8 | report_scan | If non-zero, enables scan request notification, and scan requests will be reported as events.<br>Default value: 0 |

**Table 2.208. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0b | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_bt5_set_adv_parameters_rsp_t  *gecko_cmd_le_gap_bt5_set_adv_parameters(uint8  handle,
uint16 interval_min, uint16 interval_max, uint8 channel_map, uint8 report_scan);

/* Response id */
gecko_rsp_le_gap_bt5_set_adv_parameters_id

/* Response structure */
struct gecko_msg_le_gap_bt5_set_adv_parameters_rsp_t
{
  uint16 result;
};
```

**Table 2.209. Events Generated**

| Event | Description |
|---|---|
| le_gap_scan_request | Triggered when a scan request is received during advertising if the scan request notification is enabled by this command. |

#### 2.10.1.3 (deprecated) cmd_le_gap_bt5_set_mode

**Deprecated** and replaced by le_gap_start_advertising command to start advertising, and le_gap_stop_advertising command to stop advertising. le_gap_set_advertise_timing command can be used for setting the maxevents and command le_gap_set_advertise_configuration can be used for setting address types.

**Table 2.210. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0a | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | discover | Discoverable mode |
| 6 | uint8 | connect | Connectable mode |
| 7-8 | uint16 | maxevents | If non-zero, indicates the maximum number of advertising events to send before stopping the advertiser. Value 0 indicates no maximum number limit. |
| 9 | uint8 | address_type | Address type to use for packets |

**Table 2.211. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_bt5_set_mode_rsp_t *gecko_cmd_le_gap_bt5_set_mode(uint8 handle, uint8 discover, uint8
connect, uint16 maxevents, uint8 address_type);

/* Response id */
gecko_rsp_le_gap_bt5_set_mode_id

/* Response structure */
struct gecko_msg_le_gap_bt5_set_mode_rsp_t
{
  uint16 result;
};
```

**Table 2.212. Events Generated**

| Event | Description |
|---|---|
| le_gap_adv_timeout | Triggered when the advertising events set by this command are complete and advertising is stopped on the given advertising set. |
| le_connection_opened | Triggered when a remote device opens a connection to the advertiser on the specified advertising set. |

#### 2.10.1.4 cmd_le_gap_clear_advertise_configuration

Disable advertising configuration flags on the given advertising set. The configuration change will take effect next time that advertising is enabled.

These configuration flags can be enabled using le_gap_set_advertise_configuration command.

**Table 2.213. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x13 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5-8 | uint32 | configurations | Advertising configuration flags to disable. This value can be a bit-mask of multiple flags. See le_gap_set_advertise_configuration for possible flags. |

**Table 2.214. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_gap_clear_advertise_configuration_rsp_t
*gecko_cmd_le_gap_clear_advertise_configuration(uint8 handle, uint32 configurations);

/* Response id */
gecko_rsp_le_gap_clear_advertise_configuration_id

/* Response structure */
struct gecko_msg_le_gap_clear_advertise_configuration_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.5 cmd_le_gap_clear_advertise_random_address

Clear the random address previously set for the advertiser address on an advertising set. A random address can be set using le_gap_set_advertise_random_address command. The default advertiser address will be used after this operation.

Wrong state error is returned if advertising has been enabled on the advertising set. Invalid parameter error is returned if the advertising set handle is invalid.

**Table 2.215. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x26 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |

**Table 2.216. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x26 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_gap_clear_advertise_random_address_rsp_t
*gecko_cmd_le_gap_clear_advertise_random_address(uint8 handle);

/* Response id */
gecko_rsp_le_gap_clear_advertise_random_address_id

/* Response structure */
struct gecko_msg_le_gap_clear_advertise_random_address_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.6 cmd_le_gap_connect

Connect to an advertising device with the specified initiating PHY on which connectable advertisements on primary advertising channels are received. 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 le_gap_set_discovery_timing 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 le_connection_close command with the handle received in 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 can't be established (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 an event related to the connection request. To recover from this situation, the application can implement a timeout and call le_connection_close to cancel the connection request.

This command fails with the connection limit exceeded error if the number of connections attempted exceeds the configured MAX_CONNECTIONS value.

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

Later calls of this command have to wait for the ongoing command to complete. A received event le_connection_opened indicates that the connection opened successfully and a received event le_connection_closed indicates that connection failures have occurred.

**Table 2.217. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1a | method | Message ID |
| 4-9 | bd_addr | address | Address of the device to connect to |
| 10 | uint8 | address_type | Address type of the device to connect to |
| 11 | uint8 | initiating_phy | The initiating PHY. Value:<br>• **1:** 1M PHY<br>• **4:** Coded PHY |

**Table 2.218. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | connection | Handle that will be assigned to the connection after the connection is established. This handle is valid only if the result code of this response is 0 (zero). |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_connect_rsp_t *gecko_cmd_le_gap_connect(bd_addr address, uint8 address_type, uint8
initiating_phy);

/* Response id */
gecko_rsp_le_gap_connect_id

/* Response structure */
struct gecko_msg_le_gap_connect_rsp_t
{
  uint16 result;,
  uint8 connection;
};
```

**Table 2.219. Events Generated**

| Event | Description |
|---|---|
| le_connection_opened | This event is triggered after the connection is opened and indicates whether the devices are already bonded and whether the role of the Bluetooth device is Slave or Master. |
| le_connection_parameters | This event indicates the connection parameters and security mode of the connection. |

#### 2.10.1.7 (deprecated) cmd_le_gap_discover

**Deprecated** and replaced by le_gap_start_discovery command. To preserve the same functionality when migrating to the new command, use 1M PHY in scanning_phy parameter.

This command can be used to start the GAP discovery procedure to scan for advertising devices on 1M PHY. To cancel an ongoing discovery process, use the le_gap_end_procedure command.

**Table 2.220. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | mode | Bluetooth discovery Mode. For values see link. |

**Table 2.221. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_discover_rsp_t *gecko_cmd_le_gap_discover(uint8 mode);

/* Response id */
gecko_rsp_le_gap_discover_id

/* Response structure */
struct gecko_msg_le_gap_discover_rsp_t
{
  uint16 result;
};
```

**Table 2.222. Events Generated**

| Event | Description |
|---|---|
| le_gap_scan_response | Each time an advertising packet is received, this event is triggered. The packets are not filtered in any way, so multiple events will be received for every advertising device in range. |

#### 2.10.1.8 cmd_le_gap_enable_whitelisting

Enable or disable whitelisting. The setting will be effective the next time that scanning is enabled. To add devices to the whitelist, either bond with the device or add it manually with sm_add_to_whitelist.

**Table 2.223. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x21 | method | Message ID |
| 4 | uint8 | enable | 1 enable, 0 disable whitelisting. |

**Table 2.224. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x21 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_enable_whitelisting_rsp_t *gecko_cmd_le_gap_enable_whitelisting(uint8 enable);

/* Response id */
gecko_rsp_le_gap_enable_whitelisting_id

/* Response structure */
struct gecko_msg_le_gap_enable_whitelisting_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.9 cmd_le_gap_end_procedure

End the current GAP discovery procedure (i.e., scanning for advertising devices).

**Table 2.225. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x03 | method | Message ID |

**Table 2.226. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_end_procedure_rsp_t *gecko_cmd_le_gap_end_procedure();

/* Response id */
gecko_rsp_le_gap_end_procedure_id

/* Response structure */
struct gecko_msg_le_gap_end_procedure_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.10 (deprecated) cmd_le_gap_open

**Deprecated** and replaced by le_gap_connect command, which allows opening a connection with a specified PHY.

Connect to an advertising device where 1M PHY is the initiating PHY.

**Table 2.227. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | An address of the device to connect to |
| 10 | uint8 | address_type | An address type of the device to connect to |

**Table 2.228. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | connection | A handle that will be assigned to the connection when the connection is established. The handle is valid only if the result code of this response is 0 (zero). |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_open_rsp_t *gecko_cmd_le_gap_open(bd_addr address, uint8 address_type);

/* Response id */
gecko_rsp_le_gap_open_id

/* Response structure */
struct gecko_msg_le_gap_open_rsp_t
{
  uint16 result;,
  uint8 connection;
};
```

**Table 2.229. Events Generated**

| Event | Description |
|---|---|
| le_connection_opened | Triggered after the connection is opened and indicates whether the devices are already bonded and whether the role of the Bluetooth device is Slave or Master. |

| Event | Description |
|---|---|
| le_connection_parameters | Indicates the connection parameters and security mode of the connection. |

#### 2.10.1.11 (deprecated) cmd_le_gap_set_adv_data

**Deprecated**. Use le_gap_bt5_set_adv_data command to set advertising data and scan response data.

This command is only effective on the first advertising set (handle value 0). Other advertising sets are not affected.

**Table 2.230. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | scan_rsp | This value selects if data is intended for advertising packets, scan response packets, or advertising packet in OTA. Values:<br>• **0:** Advertising packets<br>• **1:** Scan response packets<br>• **2:** OTA advertising packets<br>• **4:** OTA scan response packets |
| 5 | uint8array | adv_data | Data to be set |

**Table 2.231. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_adv_data_rsp_t *gecko_cmd_le_gap_set_adv_data(uint8 scan_rsp, uint8 adv_data_len,
const uint8 *adv_data_data);

/* Response id */
gecko_rsp_le_gap_set_adv_data_id

/* Response structure */
struct gecko_msg_le_gap_set_adv_data_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.12 (deprecated) cmd_le_gap_set_adv_parameters

**Deprecated** and replaced by le_gap_set_advertise_timing command to set the advertising intervals and le_gap_set_advertise_channel_map command to set the channel map.

This command is only effective on the first advertising set (handle value 0). Other advertising sets are not affected.

**Table 2.232. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | interval_min | Minimum advertising interval. Value in units of 0.625 ms<br>• Range: 0x20 to 0xFFFF<br>• Time range: 20 ms to 40.96 s<br>Default value: 100 ms |
| 6-7 | uint16 | interval_max | Maximum advertising interval. Value in units of 0.625 ms<br>• Range: 0x20 to 0xFFFF<br>• Time range: 20 ms to 40.96 s<br>• Note: interval_max should be bigger than interval_min<br>Default value: 200 ms |
| 8 | uint8 | channel_map | Advertising channel map, which determines which of the three channels will be used for advertising. This value is given as a bitmask. Values:<br>• **1:** Advertise on CH37<br>• **2:** Advertise on CH38<br>• **3:** Advertise on CH37 and CH38<br>• **4:** Advertise on CH39<br>• **5:** Advertise on CH37 and CH39<br>• **6:** Advertise on CH38 and CH39<br>• **7:** Advertise on all channels<br>Recommended value: 7<br>Default value: 7 |

**Table 2.233. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_adv_parameters_rsp_t  *gecko_cmd_le_gap_set_adv_parameters(uint16  interval_min,
uint16 interval_max, uint8 channel_map);

/* Response id */
gecko_rsp_le_gap_set_adv_parameters_id

/* Response structure */
struct gecko_msg_le_gap_set_adv_parameters_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.13 (deprecated) cmd_le_gap_set_adv_timeout

**Deprecated**. Use the new command le_gap_set_advertise_timing.

This command is only effective on the first advertising set (handle value 0). Other advertising sets are not affected.

**Table 2.234. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | maxevents | If non-zero, indicates the maximum number of advertising events to send before stopping advertiser. Value 0 indicates no maximum number limit. |

**Table 2.235. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_adv_timeout_rsp_t *gecko_cmd_le_gap_set_adv_timeout(uint8 maxevents);

/* Response id */
gecko_rsp_le_gap_set_adv_timeout_id

/* Response structure */
struct gecko_msg_le_gap_set_adv_timeout_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.14 cmd_le_gap_set_advertise_channel_map

Set the primary advertising channel map of the given advertising set. This setting will take effect next time that advertising is enabled.

**Table 2.236. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0f | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | channel_map | Advertising channel map which determines which of the three channels will be used for advertising. This value is given as a bit-mask. Values:<br>• **1:** Advertise on CH37<br>• **2:** Advertise on CH38<br>• **3:** Advertise on CH37 and CH38<br>• **4:** Advertise on CH39<br>• **5:** Advertise on CH37 and CH39<br>• **6:** Advertise on CH38 and CH39<br>• **7:** Advertise on all channels<br>Recommended value: 7<br>Default value: 7 |

**Table 2.237. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_le_gap_set_advertise_channel_map_rsp_t   *gecko_cmd_le_gap_set_advertise_channel_map(uint8
handle, uint8 channel_map);

/* Response id */
gecko_rsp_le_gap_set_advertise_channel_map_id

/* Response structure */
struct gecko_msg_le_gap_set_advertise_channel_map_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.15 cmd_le_gap_set_advertise_configuration

Enable advertising configuration flags on the given advertising set. The configuration change will take effect next time that advertising is enabled.

These configuration flags can be disabled using le_gap_clear_advertise_configuration command.

**Table 2.238. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x12 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5-8 | uint32 | configurations | Advertising configuration flags to enable. This value can be a bit-mask of multiple flags. Flags:<br>• **1 (Bit 0):** Use legacy advertising PDUs.<br>• **2 (Bit 1):** Omit advertiser's address from all PDUs (anonymous advertising). This flag is effective only in extended advertising.<br>• **4 (Bit 2):** Use le_gap_non_resolvable address type. Advertising must be in non-connectable mode if this configuration is enabled.<br>• **8 (Bit 3):** Include TX power in advertising packets. This flag is effective only in extended advertising.<br>Default value: 1 |

**Table 2.239. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_advertise_configuration_rsp_t *gecko_cmd_le_gap_set_advertise_configuration(uint8
handle, uint32 configurations);

/* Response id */
gecko_rsp_le_gap_set_advertise_configuration_id

/* Response structure */
struct gecko_msg_le_gap_set_advertise_configuration_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.10.1.16 cmd_le_gap_set_advertise_phy

Set advertising PHYs of the given advertising set. This setting will take effect next time that advertising is enabled. The invalid parameter error is returned if a PHY value is invalid or the device does not support a given PHY.

**Table 2.240. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x11 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | primary_phy | The PHY on which the advertising packets are transmitted on the primary advertising channel. If legacy advertising PDUs are used, 1M PHY must be used.<br>Values:<br>• **1:** Advertising PHY is 1M PHY<br>• **4:** Advertising PHY is Coded PHY<br>Default: 1 |
| 6 | uint8 | secondary_phy | The PHY on which the advertising packets are transmitted on the secondary advertising channel.<br>Values:<br>• **1:** Advertising PHY is 1M PHY<br>• **2:** Advertising PHY is 2M PHY<br>• **4:** Advertising PHY is Coded PHY<br>Default: 1 |

**Table 2.241. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_advertise_phy_rsp_t  *gecko_cmd_le_gap_set_advertise_phy(uint8  handle,  uint8
primary_phy, uint8 secondary_phy);

/* Response id */
gecko_rsp_le_gap_set_advertise_phy_id

/* Response structure */
struct gecko_msg_le_gap_set_advertise_phy_rsp_t
```

```
{
  uint16 result;
};
```

#### 2.10.1.17 cmd_le_gap_set_advertise_random_address

Set the advertiser on an advertising set to use a random address. This overrides the default advertiser address which is either the public device address programmed at production or the address written into persistent storage using system_set_identity_address command. This setting is stored in RAM only and does not change the identity address in persistent storage.

When setting a resolvable random address, the address parameter is ignored. The stack generates a private resolvable random address and set it as the advertiser address. The generated address is returned in the response.

To use the default advertiser address, remove this setting using le_gap_clear_advertise_random_address command.

Wrong state error is returned if advertising has been enabled on the advertising set. Invalid parameter error is returned if the advertising set handle is invalid or the address does not conforms to the Bluetooth specification.

**Table 2.242. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x25 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | addr_type | Address type:<br>• **1:** Static device address<br>• **2:** Private resolvable random address<br>• **3:** Private non-resolvable random address. This type can only be used for non-connectable advertising. |
| 6-11 | bd_addr | address | The random address to set. Ignore this field when setting a resolvable random address. |

**Table 2.243. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-11 | bd_addr | address_out | The resolvable random address set for the advetiser. Ignore this field when setting other types of random address. |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_gap_set_advertise_random_address_rsp_t
*gecko_cmd_le_gap_set_advertise_random_address(uint8 handle, uint8 addr_type, bd_addr address);

/* Response id */
gecko_rsp_le_gap_set_advertise_random_address_id
```

```
/* Response structure */
struct gecko_msg_le_gap_set_advertise_random_address_rsp_t
{
  uint16 result;,
  bd_addr address_out;
};
```

#### 2.10.1.18 cmd_le_gap_set_advertise_report_scan_request

Enable or disable the scan request notification of a given advertising set. This setting will take effect next time that advertising is enabled.

**Table 2.244. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x10 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | report_scan_req | If non-zero, enables scan request notification and scan requests will be reported as events.<br>Default value: 0 |

**Table 2.245. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_gap_set_advertise_report_scan_request_rsp_t
*gecko_cmd_le_gap_set_advertise_report_scan_request(uint8 handle, uint8 report_scan_req);

/* Response id */
gecko_rsp_le_gap_set_advertise_report_scan_request_id

/* Response structure */
struct gecko_msg_le_gap_set_advertise_report_scan_request_rsp_t
{
  uint16 result;
};
```

**Table 2.246. Events Generated**

| Event | Description |
|---|---|
| le_gap_scan_request | Triggered when a scan request is received during advertising if the scan request notification is enabled by this command. |

#### 2.10.1.19 cmd_le_gap_set_advertise_timing

Set the advertising timing parameters of the given advertising set. This setting will take effect next time that advertising is enabled.

**Table 2.247. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0e | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5-8 | uint32 | interval_min | Minimum advertising interval. Value in units of 0.625 ms<br>• Range: 0x20 to 0xFFFF<br>• Time range: 20 ms to 40.96 s<br>Default value: 100 ms |
| 9-12 | uint32 | interval_max | Maximum advertising interval. Value in units of 0.625 ms<br>• Range: 0x20 to 0xFFFF<br>• Time range: 20 ms to 40.96 s<br>• Note: interval_max should be bigger than interval_min<br>Default value: 200 ms |
| 13-14 | uint16 | duration | Advertising duration for this advertising set. Value 0 indicates no advertising duration limit and 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.<br>• Range: 0x0001 to 0xFFFF<br>• Time range: 10 ms to 655.35 s<br>Default value: 0 |
| 15 | uint8 | maxevents | If non-zero, indicates the maximum number of advertising events to send before the advertiser is stopped. Value 0 indicates no maximum number limit.<br>Default value: 0 |

**Table 2.248. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_advertise_timing_rsp_t *gecko_cmd_le_gap_set_advertise_timing(uint8 handle, uint32
interval_min, uint32 interval_max, uint16 duration, uint8 maxevents);

/* Response id */
gecko_rsp_le_gap_set_advertise_timing_id

/* Response structure */
struct gecko_msg_le_gap_set_advertise_timing_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.20 cmd_le_gap_set_advertise_tx_power

Limit the maximum advertising TX power on the given advertising set. If the value goes over the global value that was set using system_set_tx_power command, the global value will be the maximum limit. The maximum TX power of legacy advertising is further constrained to be less than +10 dBm. Extended advertising TX power can be +10 dBm and over if Adaptive Frequency Hopping is enabled.

This setting will take effect next time advertising is enabled.

By default, maximum advertising TX power is limited by the global value.

**Table 2.249. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1b | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5-6 | int16 | power | TX power in 0.1 dBm steps. For example, the value of 10 is 1 dBm and 55 is 5.5 dBm. |

**Table 2.250. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | int16 | set_power | The selected maximum advertising TX power |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_advertise_tx_power_rsp_t  *gecko_cmd_le_gap_set_advertise_tx_power(uint8  handle,
int16 power);

/* Response id */
gecko_rsp_le_gap_set_advertise_tx_power_id

/* Response structure */
struct gecko_msg_le_gap_set_advertise_tx_power_rsp_t
{
  uint16 result;,
  int16 set_power;
};
```

#### 2.10.1.21 (deprecated) cmd_le_gap_set_conn_parameters

**Deprecated** and replaced by le_gap_set_conn_timing_parameters command for setting timing parameters.

Set the default Bluetooth connection parameters. The configured values are valid for all subsequent connections that will be established. To change the parameters of an already established connection, use the command le_connection_set_parameters.

**Table 2.251. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | min_interval | Minimum value for the connection event interval. This must be set less than or equal to the max_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s<br>Default value: 20 ms |
| 6-7 | uint16 | max_interval | Maximum value for the connection event interval. This must be set greater than or equal to the min_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s<br>Default value: 50 ms |
| 8-9 | uint16 | latency | Slave latency, which defines how many connection intervals the slave can skip if it has no data to send<br>• Range: 0x0000 to 0x01f4<br>Default value: 0 |
| 10-11 | uint16 | timeout | Supervision timeout, which defines the time that the connection is maintained although the devices can't communicate at the currently configured connection intervals.<br>• Range: 0x000a to 0x0c80<br>• Time = Value x 10 ms<br>• Time Range: 100 ms to 32 s<br>• The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds<br>Set the supervision timeout at a value which allows communication attempts over at least a few connection intervals.<br><br>Default value: 1000 ms |

**Table 2.252. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x05 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_conn_parameters_rsp_t  *gecko_cmd_le_gap_set_conn_parameters(uint16  min_interval,
uint16 max_interval, uint16 latency, uint16 timeout);

/* Response id */
gecko_rsp_le_gap_set_conn_parameters_id

/* Response structure */
struct gecko_msg_le_gap_set_conn_parameters_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.22 cmd_le_gap_set_conn_phy

Set default preferred and accepted PHYs. PHY settings will be used for all subsequent connections. Non-preferred PHY can also be set if the remote device does not accept any of the preferred PHYs.

The parameter accepted_phy is used to specify PHYs that the stack can accept in a remotely-initiated PHY update request. A PHY update will not happen if none of the accepted PHYs are present in the request.

**NOTE:** 2M and Coded PHYs are not supported by all devices.

**Table 2.253. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | preferred_phy | Preferred PHYs. This parameter is a bitfield and multiple PHYs can be set.<br>• **0x01:** 1M PHY<br>• **0x02:** 2M PHY<br>• **0x04:** Coded PHY<br>• **0xff:** Any PHYs<br>Default: 0xff (no preference) |
| 5 | uint8 | accepted_phy | Accepted PHYs in remotely-initiated PHY update request. This parameter is a bitfield and multiple PHYs can be set.<br>• **0x01:** 1M PHY<br>• **0x02:** 2M PHY<br>• **0x04:** Coded PHY<br>• **0xff:** Any PHYs<br>Default: 0xff (all PHYs accepted) |

**Table 2.254. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_conn_phy_rsp_t  *gecko_cmd_le_gap_set_conn_phy(uint8  preferred_phy,  uint8
accepted_phy);

/* Response id */
gecko_rsp_le_gap_set_conn_phy_id
```

```
/* Response structure */
struct gecko_msg_le_gap_set_conn_phy_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.23 cmd_le_gap_set_conn_timing_parameters

Set the default Bluetooth connection parameters. The configured values are valid for all subsequent connections that will be established. To change parameters of an already established connection, use the command le_connection_set_timing_parameters.

**Table 2.255. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x22 | method | Message ID |
| 4-5 | uint16 | min_interval | Minimum value for the connection event interval. This must be set less than or equal to max_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s<br>Default value: 20 ms |
| 6-7 | uint16 | max_interval | Maximum value for the connection event interval. This must be set greater than or equal to min_interval.<br>• Time = Value x 1.25 ms<br>• Range: 0x0006 to 0x0c80<br>• Time Range: 7.5 ms to 4 s<br>Default value: 50 ms |
| 8-9 | uint16 | latency | Slave latency, which defines how many connection intervals the slave can skip if it has no data to send<br>• Range: 0x0000 to 0x01f4<br>Default value: 0 |
| 10-11 | uint16 | timeout | Supervision timeout, which defines the time that the connection is maintained although the devices can't communicate at the currently configured connection intervals.<br>• Range: 0x000a to 0x0c80<br>• Time = Value x 10 ms<br>• Time Range: 100 ms to 32 s<br>• The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds<br>Set the supervision timeout at a value which allows communication attempts over at least a few connection intervals.<br><br>Default value: 1000 ms |
| 12-13 | uint16 | min_ce_length | Minimum value for the connection event length. This must be set be less than or equal to max_ce_length.<br>• Time = Value x 0.625 ms<br>• Range: 0x0000 to 0xffff<br>Default value: 0x0000<br><br>Value is not currently used and is reserved for future. Set to 0. |
| 14-15 | uint16 | max_ce_length | Maximum value for the connection event length. This must be set greater than or equal to min_ce_length.<br>• Time = Value x 0.625 ms<br>• Range: 0x0000 to 0xffff<br>Default value: 0xffff |

**Table 2.256. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x22 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_conn_timing_parameters_rsp_t  *gecko_cmd_le_gap_set_conn_timing_parameters(uint16
min_interval, uint16 max_interval, uint16 latency, uint16 timeout, uint16 min_ce_length, uint16 max_ce_length);

/* Response id */
gecko_rsp_le_gap_set_conn_timing_parameters_id

/* Response structure */
struct gecko_msg_le_gap_set_conn_timing_parameters_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.24 cmd_le_gap_set_data_channel_classification

Specify a channel classification for data channels. This classification persists until overwritten with a subsequent command or until the system is reset.

**Table 2.257. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x19 | method | Message ID |
| 4 | 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.<br>• **0:** Channel n is bad.<br>• **1:** Channel n is unknown.<br>The rest of most significant bits are reserved for future use and shall be set to 0. At least two channels shall be marked as unknown. |

**Table 2.258. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_le_gap_set_data_channel_classification_rsp_t
*gecko_cmd_le_gap_set_data_channel_classification(uint8 channel_map_len, const uint8 *channel_map_data);

/* Response id */
gecko_rsp_le_gap_set_data_channel_classification_id

/* Response structure */
struct gecko_msg_le_gap_set_data_channel_classification_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.25 cmd_le_gap_set_discovery_extended_scan_response

Enable or disable the extended scan response event. When the extended scan response event is enabled, it replaces le_gap_scan_response, that is, the stack will generate either le_gap_extended_scan_response or le_gap_scan_response, but not both.

**Table 2.259. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1c | method | Message ID |
| 4 | uint8 | enable | Values:<br>• **0:** Disable extended scan response event<br>• **1:** Enable extended scan response event |

**Table 2.260. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                       gecko_msg_le_gap_set_discovery_extended_scan_response_rsp_t
*gecko_cmd_le_gap_set_discovery_extended_scan_response(uint8 enable);

/* Response id */
gecko_rsp_le_gap_set_discovery_extended_scan_response_id

/* Response structure */
struct gecko_msg_le_gap_set_discovery_extended_scan_response_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.26 cmd_le_gap_set_discovery_timing

Set the timing parameters of scanning on the specified PHYs. If the device is currently scanning for advertising devices on PHYs, new parameters will take effect when scanning is restarted.

**Table 2.261. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x16 | method | Message ID |
| 4 | uint8 | phys | PHYs for which the parameters are set.<br>• **1:** 1M PHY<br>• **4:** Coded PHY<br>• **5:** 1M PHY and Coded PHY |
| 5-6 | uint16 | scan_interval | Scan interval is defined as the time interval when the device starts its last scan until it begins the subsequent scan. In other words, how often to scan<br>• Time = Value x 0.625 ms<br>• Range: 0x0004 to 0xFFFF<br>• Time Range: 2.5 ms to 40.96 s<br>Default value: 10 ms<br><br>A variable delay occurs when switching channels at the end of each scanning interval, which is included in the scanning interval time. During the switch time, advertising packets are not received by the device. The switch time variation is use case-dependent. For example, if scanning while keeping active connections, the channel switch time might be longer than when scanning without any active connections. Increasing the scanning interval reduces the amount of time in which the device can't receive advertising packets because it switches channels less often.<br><br>After every scan interval, the scanner changes the frequency at which it operates. It cycles through all three advertising channels in a round robin fashion. According to the specification, all three channels must be used by a scanner. |
| 7-8 | uint16 | scan_window | Scan window defines the duration of the scan which must be less than or equal to the scan_interval<br>• Time = Value x 0.625 ms<br>• Range: 0x0004 to 0xFFFF<br>• Time Range: 2.5 ms to 40.96 s<br>Default value: 10 ms Note that the packet reception is aborted if it's started just before the scan window ends. |

**Table 2.262. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x16 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_discovery_timing_rsp_t *gecko_cmd_le_gap_set_discovery_timing(uint8 phys, uint16
scan_interval, uint16 scan_window);

/* Response id */
gecko_rsp_le_gap_set_discovery_timing_id

/* Response structure */
struct gecko_msg_le_gap_set_discovery_timing_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.27 cmd_le_gap_set_discovery_type

Set the scan type on the specified PHYs. If the device is currently scanning for advertising devices on PHYs, new parameters will take effect when scanning is restarted.

**Table 2.263. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x17 | method | Message ID |
| 4 | uint8 | phys | PHYs for which the parameters are set.<br>• **1:** 1M PHY<br>• **4:** Coded PHY<br>• **5:** 1M PHY and Coded PHY |
| 5 | uint8 | scan_type | Scan type. Values:<br>• **0:** Passive scanning<br>• **1:** Active scanning<br>• In passive scanning mode, the device only listens to advertising packets and does not transmit packets.<br>• In active scanning mode, the device sends out a scan request packet upon receiving an advertising packet from a remote device. Then, it listens to the scan response packet from the remote device.<br>Default value: 0 |

**Table 2.264. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_discovery_type_rsp_t  *gecko_cmd_le_gap_set_discovery_type(uint8  phys,  uint8
scan_type);

/* Response id */
gecko_rsp_le_gap_set_discovery_type_id

/* Response structure */
struct gecko_msg_le_gap_set_discovery_type_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.28 cmd_le_gap_set_long_advertising_data

Set advertising data for a specified packet type and advertising set. Data currently in the system data buffer will be extracted as the advertising data. The buffer will be emptied after this command regardless of the completion status.

Prior to calling this command, add data to the buffer with one or multiple calls of system_data_buffer_write.

Maximum 31 bytes of data can be set for legacy advertising. Maximum 191 bytes of data can be set for connectable extended advertising. Maximum 1650 bytes of data can be set for periodic and non-connectable extended advertising, but advertising parameters may limit the amount of data that can be sent in a single advertisement.

See le_gap_bt5_set_adv_data for more details on advertising data.

**Table 2.265. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x20 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5 | uint8 | packet_type | This value selects whether data is intended for advertising packets, scan response packets, or periodic advertising packets. Values:<br>• **0:** Advertising packets<br>• **1:** Scan response packets<br>• **2:** OTA advertising packets<br>• **4:** OTA scan response packets<br>• **8:** Periodic advertising packets |

**Table 2.266. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x20 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_long_advertising_data_rsp_t  *gecko_cmd_le_gap_set_long_advertising_data(uint8
handle, uint8 packet_type);

/* Response id */
gecko_rsp_le_gap_set_long_advertising_data_id

/* Response structure */
struct gecko_msg_le_gap_set_long_advertising_data_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.10.1.29 (deprecated) cmd_le_gap_set_mode

**Deprecated**. Use le_gap_start_advertising command to enable advertising and le_gap_stop_advertising command to disable advertising.

This command is only effective on the first advertising set (handle value 0). Other advertising sets are not affected.

**Table 2.267. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | discover | Discoverable mode |
| 5 | uint8 | connect | Connectable mode |

**Table 2.268. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_mode_rsp_t *gecko_cmd_le_gap_set_mode(uint8 discover, uint8 connect);

/* Response id */
gecko_rsp_le_gap_set_mode_id

/* Response structure */
struct gecko_msg_le_gap_set_mode_rsp_t
{
  uint16 result;
};
```

**Table 2.269. Events Generated**

| Event | Description |
|---|---|
| le_gap_adv_timeout | Triggered when the number of advertising events is done and advertising has stopped. |
| le_connection_opened | Triggered when a remote device opens a connection to this advertising device. |

#### 2.10.1.30 cmd_le_gap_set_privacy_mode

Enable or disable the privacy feature on all GAP roles. New privacy mode will take effect for advertising next time advertising is enabled, for scanning next time scanning is enabled, 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 a 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.

**Table 2.270. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0d | method | Message ID |
| 4 | uint8 | privacy | Values:<br>• **0:** Disable privacy<br>• **1:** Enable privacy |
| 5 | uint8 | interval | The minimum time interval between a private address change. This parameter is ignored if this command is issued to disable privacy mode. Values:<br>• **0:** Use default interval, 15 minutes<br>• **others:** The time interval in minutes |

**Table 2.271. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_set_privacy_mode_rsp_t  *gecko_cmd_le_gap_set_privacy_mode(uint8  privacy,  uint8
interval);

/* Response id */
gecko_rsp_le_gap_set_privacy_mode_id

/* Response structure */
struct gecko_msg_le_gap_set_privacy_mode_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.31 (deprecated) cmd_le_gap_set_scan_parameters

**Deprecated** and replaced by le_gap_set_discovery_timing command to set timing parameters, and le_gap_set_discovery_type command for the scan type.

The parameters set by this command are only effective on the 1M PHY. For Coded PHY, use the above replacement command.

**Table 2.272. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | scan_interval | Scanner interval is defined as the time interval when the device starts its last scan until it begins the subsequent scan. In other words, it indicates how often to scan<br>• Time = Value x 0.625 ms<br>• Range: 0x0004 to 0x4000<br>• Time Range: 2.5 ms to 10.24 s<br>Default value: 10 ms<br><br>A variable delay occurs when switching channels at the end of each scanning interval, which is included in the scanning interval time. During the switch time no advertising packets are received by the device. The switch time variation is use case-dependent. For example, if scanning while keeping active connections, the channel switch time might be longer than scanning without any active connections. Increasing the scanning interval reduces the amount of time in which the device can't receive advertising packets because it will switch channels less often.<br><br>After every scan interval, the scanner changes the frequency at which it operates. It cycles through all three advertising channels in a round robin fashion. According to the specification, all three channels must be used by the scanner. |
| 6-7 | uint16 | scan_window | Scan window defines the duration of the scan which must be less than or equal to scan_interval<br>• Time = Value x 0.625 ms<br>• Range: 0x0004 to 0x4000<br>• Time Range: 2.5 ms to 10.24 s<br>Default value: 10 ms Note that packet reception is aborted if it was started before the scan window ends. |
| 8 | uint8 | active | The scan type. Values:<br>• **0:** Passive scanning<br>• **1:** Active scanning<br>• In passive scanning mode, the device only listens to advertising packets and does not transmit any packets.<br>• In active scanning mode, the device will send out a scan request packet upon receiving an advertising packet from a remote device and then it will listen to the scan response packet from the device.<br>Default value: 0 |

**Table 2.273. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_set_scan_parameters_rsp_t *gecko_cmd_le_gap_set_scan_parameters(uint16 scan_interval,
uint16 scan_window, uint8 active);

/* Response id */
gecko_rsp_le_gap_set_scan_parameters_id

/* Response structure */
struct gecko_msg_le_gap_set_scan_parameters_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.32 cmd_le_gap_start_advertising

Start advertising of a given advertising set with specified discoverable and connectable modes.

The number of concurrent advertising is limited by MAX_ADVERTISERS configuration.

The number of concurrent connectable advertising 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 1M PHY. The stack will automatically select extended advertising PDUs if either of the following has occurred with the default configuration:

1. The connectable mode is set to le_gap_connectable_non_scannable.
2. The primary advertising PHY is set to Coded PHY by the command le_gap_set_advertise_phy.
3. The user advertising data length is more than 31 bytes.
4. Periodic advertising is enabled.

If currently set parameters can't be used, an error is returned. Specifically, this command fails with the connection limit exceeded error if it causes the number of connections exceeding the configured MAX_CONNECTIONS value. It fails with the invalid parameter error if one of the following use cases occurs:

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

If advertising is enabled in user_data mode, use le_gap_bt5_set_adv_data to set advertising and scan response data before issuing this command. When advertising is enabled in modes other than user_data, 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 the 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 the device is not in privacy mode. If 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.

Event le_connection_opened will be received when a remote device opens a connection to the advertiser on this advertising set and also advertising on the given set stops.

Event le_gap_adv_timeout will be received when the number of advertising events set by command is done and advertising with the current set has stopped.

**Table 2.274. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x14 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |

| Byte | Type | Name | Description |
|---|---|---|---|
| 5 | uint8 | discover | Discoverable mode |
| 6 | uint8 | connect | Connectable mode |

**Table 2.275. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_start_advertising_rsp_t  *gecko_cmd_le_gap_start_advertising(uint8  handle,  uint8
discover, uint8 connect);

/* Response id */
gecko_rsp_le_gap_start_advertising_id

/* Response structure */
struct gecko_msg_le_gap_start_advertising_rsp_t
{
  uint16 result;
};
```

**Table 2.276. Events Generated**

| Event | Description |
|---|---|
| le_gap_adv_timeout | Triggered when the number of advertising events set by command is done and advertising has stopped on the given advertising set. |
| le_connection_opened | Triggered when a remote device opens a connection to the advertiser on the specified advertising set and also advertising with the current set stops. |

#### 2.10.1.33 cmd_le_gap_start_discovery

Start the GAP discovery procedure to scan for advertising devices on the specified scanning PHY or to perform a device discovery. To cancel an ongoing discovery process use the le_gap_end_procedure command.

The invalid parameter error will be returned if the scanning PHY value is invalid or the device does not support the PHY.

**Table 2.277. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x18 | method | Message ID |
| 4 | uint8 | scanning_phy | The scanning PHY. Value:<br>• **1:** 1M PHY<br>• **4:** Coded PHY |
| 5 | uint8 | mode | Bluetooth discovery Mode. For values see link. |

**Table 2.278. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_start_discovery_rsp_t  *gecko_cmd_le_gap_start_discovery(uint8  scanning_phy,  uint8
mode);

/* Response id */
gecko_rsp_le_gap_start_discovery_id

/* Response structure */
struct gecko_msg_le_gap_start_discovery_rsp_t
{
  uint16 result;
};
```

**Table 2.279. Events Generated**

| Event | Description |
|---|---|
| le_gap_scan_response | This event is triggered each time an advertising packet is received. Packets are not filtered in any way, so multiple events will be received for every advertising device in range. |
| le_gap_extended_scan_response | Each time an advertising packet is received and extended scan response is enabled (by le_gap_set_discovery_extended_scan_response), this event is triggered. The packets are not filtered in any way. As a result, multiple events will be received for every advertising device in range. |

#### 2.10.1.34 cmd_le_gap_start_periodic_advertising

Start periodic advertising on the given advertising set. The stack enables the advertising set automatically if the set was not enabled and the set can advertise using extended advertising PDUs beside the syncInfo (which is needed for the periodic advertising).

The invalid parameter error is returned if the application has configured legacy advertising PDUs or anonymous advertising, or the advertising set is enabled using legacy advertising PDUs.

**Table 2.280. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1d | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |
| 5-6 | uint16 | interval_min | Minimum periodic advertising interval. Value in units of 1.25 ms<br>• Range: 0x06 to 0xFFFF<br>• Time range: 7.5 ms to 81.92 s<br>Default value: 100 ms |
| 7-8 | uint16 | interval_max | Maximum periodic advertising interval. Value in units of 1.25 ms<br>• Range: 0x06 to 0xFFFF<br>• Time range: 7.5 ms to 81.92 s<br>• Note: interval_max should be bigger than interval_min<br>Default value: 200 ms |
| 9-12 | uint32 | flags | Periodic advertising configurations. Bitmask of the following:<br>• **Bit 0:** Include TX power in advertising PDU |

**Table 2.281. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_le_gap_start_periodic_advertising_rsp_t  *gecko_cmd_le_gap_start_periodic_advertising(uint8
handle, uint16 interval_min, uint16 interval_max, uint32 flags);

/* Response id */
gecko_rsp_le_gap_start_periodic_advertising_id

/* Response structure */
struct gecko_msg_le_gap_start_periodic_advertising_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.10.1.35 cmd_le_gap_stop_advertising

Stop the advertising of the given advertising set.

**Table 2.282. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x15 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |

**Table 2.283. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_le_gap_stop_advertising_rsp_t *gecko_cmd_le_gap_stop_advertising(uint8 handle);

/* Response id */
gecko_rsp_le_gap_stop_advertising_id

/* Response structure */
struct gecko_msg_le_gap_stop_advertising_rsp_t
{
  uint16 result;
};
```

#### 2.10.1.36 cmd_le_gap_stop_periodic_advertising

Stop the periodic advertising on the given advertising set.

This command does not affect the enable state of the advertising set, i.e., legacy or extended advertising is not stopped.

**Table 2.284. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1f | method | Message ID |
| 4 | uint8 | handle | Advertising set handle |

**Table 2.285. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x1f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_le_gap_stop_periodic_advertising_rsp_t   *gecko_cmd_le_gap_stop_periodic_advertising(uint8
handle);

/* Response id */
gecko_rsp_le_gap_stop_periodic_advertising_id

/* Response structure */
struct gecko_msg_le_gap_stop_periodic_advertising_rsp_t
{
  uint16 result;
};
```

### 2.10.2 le_gap events

#### 2.10.2.1 evt_le_gap_adv_timeout

Indicates that the advertiser has completed the configured number of advertising events in the advertising set and advertising has stopped. The maximum number of advertising events can be configured by the maxevents parameter in the command le_gap_set_advertise_timing.

**Table 2.286. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | handle | The advertising set handle |

**C Functions**

```
/* Event id */
gecko_evt_le_gap_adv_timeout_id

/* Event structure */
struct gecko_msg_le_gap_adv_timeout_evt_t
{
  uint8 handle;
};
```

#### 2.10.2.2 evt_le_gap_extended_scan_response

Reports an advertising or scan response packet that is received by the device's radio while in scanning mode.

By default, this event is disabled and the stack will not generate it. The application needs to enable it using le_gap_set_discovery_extended_scan_response command. When this event is enabled, it replaces le_gap_scan_response, that is, the stack will generate either this event or le_gap_scan_response, but not both.

**Table 2.287. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | packet_type | **Bits 0..2**: advertising packet type<br>• **000**: Connectable scannable undirected advertising<br>• **001**: Connectable undirected advertising<br>• **010**: Scannable undirected advertising<br>• **011**: Non-connectable non-scannable undirected advertising<br>• **100**: Scan Response. Note that this is received only if the device is in active scan mode.<br>**Bits 3..4**: Reserved for future<br>**Bits 5..6**: data completeness<br>• **00:** Complete<br>• **01:** Incomplete, more data to come in new events<br>• **10:** Incomplete, data truncated, no more to come<br>**Bit 7**: legacy or extended advertising<br>• **0:** Legacy advertising PDUs used<br>• **1:** Extended advertising PDUs used |
| 5-10 | bd_addr | address | Bluetooth address of the remote device |
| 11 | uint8 | address_type | Advertiser address type. Values:<br>• **0:** Public address<br>• **1:** Random address<br>• **255:** No address provided (anonymous advertising) |
| 12 | uint8 | bonding | Bonding handle if the remote advertising device has previously bonded with the local device. Values:<br>• **0xff:** No bonding<br>• **Other:** Bonding handle |
| 13 | uint8 | primary_phy | The PHY on which advertising packets are transmitted on the primary advertising channel.<br>Values:<br>• **1:** 1M PHY<br>• **4:** Coded PHY |

| Byte | Type | Name | Description |
|---|---|---|---|
| 14 | uint8 | secondary_phy | The PHY on which advertising packets are transmitted on the secondary advertising channel.<br>Values:<br>• **1:** Advertising PHY is 1M PHY<br>• **2:** Advertising PHY is 2M PHY<br>• **4:** Advertising PHY is Coded PHY |
| 15 | uint8 | adv_sid | Advertising set identifier |
| 16 | int8 | tx_power | TX power value in the received packet header. Units: dBm<br>• Valid value range: -127 to 126<br>• Value 127: information unavailable |
| 17 | int8 | rssi | Signal strength indicator (RSSI) in the last received packet. Units: dBm<br>• Range: -127 to +20 |
| 18 | uint8 | channel | The channel number on which the last packet was received |
| 19-20 | uint16 | periodic_interval | The periodic advertising interval. Value 0 indicates no periodic advertising. Otherwise,<br>• Range: 0x06 to 0xFFFF<br>• Unit: 1.25 ms<br>• Time range: 7.5 ms to 81.92 s |
| 21 | uint8array | data | Advertising or scan response data |

**C Functions**

```
/* Event id */
gecko_evt_le_gap_extended_scan_response_id

/* Event structure */
struct gecko_msg_le_gap_extended_scan_response_evt_t
{
  uint8 packet_type;,
  bd_addr address;,
  uint8 address_type;,
  uint8 bonding;,
  uint8 primary_phy;,
  uint8 secondary_phy;,
  uint8 adv_sid;,
  int8 tx_power;,
  int8 rssi;,
  uint8 channel;,
  uint16 periodic_interval;,
  uint8array data;
};
```

#### 2.10.2.3 evt_le_gap_scan_request

Reports any scan request received in advertising mode if the scan request notification is enabled. Do not confuse this event with the scan response.

**Table 2.288. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | handle | Advertising set handle where scan request was received |
| 5-10 | bd_addr | address | Bluetooth address of the scanning device |
| 11 | uint8 | address_type | Scanner address type. Values:<br>• **0:** Public address<br>• **1:** Random address |
| 12 | uint8 | bonding | Bonding handle if the remote scanning device has previously bonded with the local device. Values:<br>• **0xff:** No bonding<br>• **Other:** Bonding handle |

**C Functions**

```
/* Event id */
gecko_evt_le_gap_scan_request_id

/* Event structure */
struct gecko_msg_le_gap_scan_request_evt_t
{
  uint8 handle;,
  bd_addr address;,
  uint8 address_type;,
  uint8 bonding;
};
```

#### 2.10.2.4 evt_le_gap_scan_response

Reports any advertising or scan response packet that is received by the device's radio while in scanning mode.

Note that this event will be replaced by le_gap_extended_scan_response if extended scan response event is enabled. The extended scan response event can be enabled or disabled using command le_gap_set_discovery_extended_scan_response command.

**Table 2.289. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile |
| 3 | 0x00 | method | Message ID |
| 4 | int8 | rssi | Signal strength indicator (RSSI) in the latest received packet. Units: dBm<br>• Range: -127 to +20 |
| 5 | uint8 | packet_type | **Bits 0..2**: advertising packet type<br>• **000**: Connectable scannable undirected advertising<br>• **001**: Connectable undirected advertising<br>• **010**: Scannable undirected advertising<br>• **011**: Non-connectable non-scannable undirected advertising<br>• **100**: Scan Response. Note that this is received only if the device is in active scan mode.<br>**Bits 3..4**: Reserved for the future<br>**Bits 5..6**: data completeness<br>• **00:** Complete<br>• **01:** Incomplete, more data to come in new events<br>• **10:** Incomplete, data truncated, no more to come<br>**Bit 7**: legacy or extended advertising<br>• **0:** Legacy advertising PDUs used<br>• **1:** Extended advertising PDUs used |
| 6-11 | bd_addr | address | Bluetooth address of the remote device |
| 12 | uint8 | address_type | Advertiser address type. Values:<br>• **0:** Public address<br>• **1:** Random address<br>• **255:** No address provided (anonymous advertising) |
| 13 | uint8 | bonding | Bonding handle if the remote advertising device has previously bonded with the local device. Values:<br>• **0xff:** No bonding<br>• **Other:** Bonding handle |
| 14 | uint8array | data | Advertising or scan response data |

**C Functions**

```
/* Event id */
gecko_evt_le_gap_scan_response_id

/* Event structure */
struct gecko_msg_le_gap_scan_response_evt_t
{
```

```
  int8 rssi;,
  uint8 packet_type;,
  bd_addr address;,
  uint8 address_type;,
  uint8 bonding;,
  uint8array data;
};
```

### 2.10.3 le_gap enumerations

#### 2.10.3.1 enum_le_gap_address_type

These values define the Bluetooth Address types used by the stack.

**Table 2.290. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | le_gap_address_type_public | Public address |
| 1 | le_gap_address_type_random | Random address |

#### 2.10.3.2 enum_le_gap_adv_address_type

**Deprecated**. This can only be used in deprecated command le_gap_bt5_set_mode.

Address type to use for starting advertising with le_gap_bt5_set_mode.

**Table 2.291. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | le_gap_identity_address | Use public or static device address, or an identity address if privacy mode is enabled. |
| 1 | le_gap_non_resolvable | Use non resolvable address type; advertising mode must also be non-connectable. |

#### 2.10.3.3 enum_le_gap_connectable_mode

These values define the available connectable modes, which indicate whether the device accepts connection requests or scan requests.

**Table 2.292. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | le_gap_non_connectable | Non-connectable non-scannable. |
| 1 | le_gap_directed_connectable | Directed connectable (RESERVED, DO NOT USE) |
| 2 | le_gap_undirected_connectable | Undirected connectable scannable.<br>**Deprecated**, replaced by enumeration le_gap_connectable_scannable.<br>This mode can only be used in legacy advertising PDUs. |
| 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. |

#### 2.10.3.4 enum_le_gap_discover_mode

These values indicate which Bluetooth discovery mode to use when scanning for advertising devices.

**Table 2.293. Enumerations**

| Value | Name | Description |
|---|---|---|
| 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. |

#### 2.10.3.5 enum_le_gap_discoverable_mode

These values define the available Discoverable Modes, which dictate how the device is visible to other devices.

**Table 2.294. Enumerations**

| Value | Name | Description |
|---|---|---|
| 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 using the Observation procedure. |
| 4 | le_gap_user_data | Send advertising and/or scan response data defined by the user with le_gap_bt5_set_adv_data. The limited/general discoverable flags are defined by the user. |

#### 2.10.3.6 enum_le_gap_phy_type

Types of PHYs used within le_gap class

**Table 2.295. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | le_gap_phy_1m | 1M PHY |
| 2 | le_gap_phy_2m | 2M PHY |
| 4 | le_gap_phy_coded | Coded PHY |

## 2.11 Bluetooth Mesh Configuration Client (mesh_config_client)

Bluetooth mesh stack API for the Mesh Configuration Client

Commands in this class configure nodes in the Mesh network, which includes key management, publish and subscribe settings manipulation, and node feature configuration.

Requests to nodes are asynchronous. A handle is assigned to each request that is pending a reply from a node in the network. The handle can be used to query the request status, and to identify the response event from the node. Multiple requests can be made in parallel (as long as they are destined to different nodes; only one pending request per node is allowed).

**Request Management**

- Cancel a request
- Query current status of a request
- Get default request timeout
- Set default request timeout

**Key and Mesh Network Management**

- Deploy a network key to a node
- Remove a network key from a node
- List network keys on a node
- Deploy an application key to a node
- Remove an application key from a node
- List application keys bound to a network key on a node

**Node Configuration**

- Get device composition data of a node
- Reset a node
- Get node default TTL state value
- Set node default TTL state value
- Get node secure network beacon state value
- Set node secure network beacon state value
- Get node identity advertising state value
- Set node identity advertising state value
- Get node friend state value
- Set node friend state value
- Get node LPN poll timeout state value
- Get node GATT proxy state value
- Set node GATT proxy state value
- Get node relay state value
- Set node relay state value
- Get node network transmit state value
- Set node network transmit state value

**Model Configuration**

- Bind a model to an application key
- Remove a model to application key binding
- List model to application key bindings on a node
- Add a subscription address to a model
- 
- Remove a subscription address from a model
- 
- Overwrite the subscription list of a model with an address
- 
- Clear the subscription list of a model
- Get the subscription list of a model
- Get a model's publication parameters
- Set a model's publication parameters

**Heartbeat**

- Get node heartbeat publication settings
- Set node heartbeat publication settings
- Get node heartbeat subscription settings
- Set node heartbeat subscription settings

### 2.11.1 mesh_config_client commands

#### 2.11.1.1 cmd_mesh_config_client_add_appkey

Add an application key to a node.

**Table 2.296. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | appkey_index | Index of the application key to add |
| 10-11 | uint16 | netkey_index | Index of the network key to bind the application key to on the node. Note that this may be different from the binding on other nodes or on the Configuration Client if desired. |

**Table 2.297. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_add_appkey_rsp_t    *gecko_cmd_mesh_config_client_add_appkey(uint16
enc_netkey_index, uint16 server_address, uint16 appkey_index, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_config_client_add_appkey_id

/* Response structure */
struct gecko_msg_mesh_config_client_add_appkey_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.298. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_appkey_status | This event is created when a response for an add application key or a remove application key request is received or the request times out. |

#### 2.11.1.2 cmd_mesh_config_client_add_model_sub

Add an address to model subscription list.

**Table 2.299. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-14 | uint16 | sub_address | The address to add to the subscription list. Note that the address has to be a group address. |

**Table 2.300. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_add_model_sub_rsp_t  *gecko_cmd_mesh_config_client_add_model_sub(uint16
enc_netkey_index,  uint16  server_address,  uint8  element_index,  uint16  vendor_id,  uint16  model_id,  uint16
sub_address);

/* Response id */
gecko_rsp_mesh_config_client_add_model_sub_id

/* Response structure */
struct gecko_msg_mesh_config_client_add_model_sub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.301. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.3 cmd_mesh_config_client_add_model_sub_va

Add a Label UUID (full virtual address) to model subscription list

**Table 2.302. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x19 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-28 | uuid_128 | sub_address | The full virtual address to add to the subscription list |

**Table 2.303. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_add_model_sub_va_rsp_t
*gecko_cmd_mesh_config_client_add_model_sub_va(uint16   enc_netkey_index,   uint16   server_address,   uint8
element_index, uint16 vendor_id, uint16 model_id, uuid_128 sub_address);

/* Response id */
gecko_rsp_mesh_config_client_add_model_sub_va_id

/* Response structure */
struct gecko_msg_mesh_config_client_add_model_sub_va_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.304. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.4 cmd_mesh_config_client_add_netkey

Add a network key to a node.

**Table 2.305. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | netkey_index | Index of the network key to add |

**Table 2.306. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_add_netkey_rsp_t    *gecko_cmd_mesh_config_client_add_netkey(uint16
enc_netkey_index, uint16 server_address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_config_client_add_netkey_id

/* Response structure */
struct gecko_msg_mesh_config_client_add_netkey_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.307. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_netkey_status | This event is created when a response for an add network key or a remove network key request is received, or the request times out. |

#### 2.11.1.5 cmd_mesh_config_client_bind_model

Bind an application key to a model.

**Table 2.308. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | appkey_index | Index of the application key to bind to the model |
| 11-12 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 13-14 | uint16 | model_id | Model ID for the model to configure |

**Table 2.309. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_bind_model_rsp_t    *gecko_cmd_mesh_config_client_bind_model(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 appkey_index, uint16 vendor_id, uint16
model_id);

/* Response id */
gecko_rsp_mesh_config_client_bind_model_id

/* Response structure */
struct gecko_msg_mesh_config_client_bind_model_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.310. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_binding_status | Status event for binding and unbinding application keys and models. |

#### 2.11.1.6 cmd_mesh_config_client_cancel_request

Cancel an ongoing request releasing resources allocated at the Configuration Client. Note that this call does no undo any setting a node may have made if it had received the request already.

**Table 2.311. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |

**Table 2.312. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_cancel_request_rsp_t  *gecko_cmd_mesh_config_client_cancel_request(uint32
handle);

/* Response id */
gecko_rsp_mesh_config_client_cancel_request_id

/* Response structure */
struct gecko_msg_mesh_config_client_cancel_request_rsp_t
{
  uint16 result;
};
```

#### 2.11.1.7 cmd_mesh_config_client_clear_model_sub

Clear (empty) the model subscription address list.

**Table 2.313. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |

**Table 2.314. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_config_client_clear_model_sub_rsp_t *gecko_cmd_mesh_config_client_clear_model_sub(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_config_client_clear_model_sub_id

/* Response structure */
struct gecko_msg_mesh_config_client_clear_model_sub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.315. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.8 cmd_mesh_config_client_get_beacon

Get node secure network beacon state.

**Table 2.316. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.317. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_get_beacon_rsp_t    *gecko_cmd_mesh_config_client_get_beacon(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_beacon_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_beacon_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.318. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_beacon_status | Status event for get beacon state and set beacon state commands. |

#### 2.11.1.9 cmd_mesh_config_client_get_dcd

Get composition data of a device.

**Table 2.319. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2c | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | page | Composition data page to query |

**Table 2.320. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct      gecko_msg_mesh_config_client_get_dcd_rsp_t      *gecko_cmd_mesh_config_client_get_dcd(uint16
enc_netkey_index, uint16 server_address, uint8 page);

/* Response id */
gecko_rsp_mesh_config_client_get_dcd_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_dcd_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.321. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_dcd_data | Event reporting queried composition data page contents. The contents are requested using the get device composition data command. More than one event may be generated. Page contents are terminated by a composition data end event. Note that the interpretation of the received data is page-specific. Page 0 contains the element and model layout of the node. |
| mesh_config_client_dcd_data_end | Terminating event for node composition data |

#### 2.11.1.10 cmd_mesh_config_client_get_default_timeout

Get the default timeout for configuration client requests. If there is no response when the timeout expires, a configuration request is considered to have failed and an event with an error result will be generated. Note that if the Bluetooth mesh stack notices the request is destined to an LPN by receiving an on-behalf-of acknowledgment from a Friend node, the timeout in use will be changed to the LPN default timeout.

**Table 2.322. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2e | method | Message ID |

**Table 2.323. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | timeout_ms | Timeout in milliseconds. Default timeout is 5 s (5000 ms). |
| 10-13 | uint32 | lpn_timeout_ms | Timeout in milliseconds when communicating with an LPN node. Default LPN timeout is 120 s (120000 ms). |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_get_default_timeout_rsp_t
*gecko_cmd_mesh_config_client_get_default_timeout();

/* Response id */
gecko_rsp_mesh_config_client_get_default_timeout_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_default_timeout_rsp_t
{
  uint16 result;,
  uint32 timeout_ms;,
  uint32 lpn_timeout_ms;
};
```

#### 2.11.1.11 cmd_mesh_config_client_get_default_ttl

Get node default TTL state.

**Table 2.324. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1d | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.325. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_config_client_get_default_ttl_rsp_t *gecko_cmd_mesh_config_client_get_default_ttl(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_default_ttl_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_default_ttl_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.326. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_default_ttl_status | Status event for get default TTL state and set default TTL state commands. |

#### 2.11.1.12 cmd_mesh_config_client_get_friend

Get node friend state.

**Table 2.327. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x27 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.328. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x27 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_get_friend_rsp_t    *gecko_cmd_mesh_config_client_get_friend(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_friend_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_friend_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.329. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_friend_status | Status event for get friend state and set friend state commands. |

#### 2.11.1.13 cmd_mesh_config_client_get_gatt_proxy

Get node GATT proxy state.

**Table 2.330. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1f | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.331. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_get_gatt_proxy_rsp_t  *gecko_cmd_mesh_config_client_get_gatt_proxy(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_gatt_proxy_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_gatt_proxy_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.332. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_gatt_proxy_status | Status event for get GATT proxy state and set GATT proxy state commands |

#### 2.11.1.14 cmd_mesh_config_client_get_heartbeat_pub

Get the heartbeat publication state of a node.

**Table 2.333. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.334. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_get_heartbeat_pub_rsp_t
*gecko_cmd_mesh_config_client_get_heartbeat_pub(uint16 enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_heartbeat_pub_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_heartbeat_pub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.335. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_heartbeat_pub_status | Status event for get heartbeat publication state and set heartbeat publication state commands |

#### 2.11.1.15 cmd_mesh_config_client_get_heartbeat_sub

Get the heartbeat subscription state of a node.

**Table 2.336. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.337. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_get_heartbeat_sub_rsp_t
*gecko_cmd_mesh_config_client_get_heartbeat_sub(uint16 enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_heartbeat_sub_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_heartbeat_sub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.338. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_heartbeat_sub_status | Status event for get heartbeat subscription state and set heartbeat subscription state commands |

#### 2.11.1.16 cmd_mesh_config_client_get_identity

Get node identity state.

**Table 2.339. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | netkey_index | Network key index for which the state is queried |

**Table 2.340. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_config_client_get_identity_rsp_t   *gecko_cmd_mesh_config_client_get_identity(uint16
enc_netkey_index, uint16 server_address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_config_client_get_identity_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_identity_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.341. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_identity_status | Status event for get node identity state and set node identity state commands. |

#### 2.11.1.17 cmd_mesh_config_client_get_lpn_polltimeout

Get the LPN poll timeout from a Friend node.

**Table 2.342. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2b | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | lpn_address | LPN address |

**Table 2.343. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_get_lpn_polltimeout_rsp_t
*gecko_cmd_mesh_config_client_get_lpn_polltimeout(uint16  enc_netkey_index,  uint16  server_address,  uint16
lpn_address);

/* Response id */
gecko_rsp_mesh_config_client_get_lpn_polltimeout_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_lpn_polltimeout_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.344. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_lpn_polltimeout_status | Status event for get LPN poll timeout command. |

#### 2.11.1.18 cmd_mesh_config_client_get_model_pub

Get model publication state.

**Table 2.345. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be queried resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to query. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to query |

**Table 2.346. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_get_model_pub_rsp_t  *gecko_cmd_mesh_config_client_get_model_pub(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_config_client_get_model_pub_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_model_pub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.347. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_pub_status | Status event for get model publication state, set model publication state, commands. |

#### 2.11.1.19 cmd_mesh_config_client_get_network_transmit

Get node network transmit state.

**Table 2.348. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x23 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.349. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x23 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_get_network_transmit_rsp_t
*gecko_cmd_mesh_config_client_get_network_transmit(uint16 enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_network_transmit_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_network_transmit_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.350. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_network_transmit_status | Status event for get network transmit state and set network transmit state commands |

#### 2.11.1.20 cmd_mesh_config_client_get_relay

Get node relay state.

**Table 2.351. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x21 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.352. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x21 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct     gecko_msg_mesh_config_client_get_relay_rsp_t     *gecko_cmd_mesh_config_client_get_relay(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_get_relay_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_relay_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.353. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_relay_status | Status event for get relay state and set relay state commands |

#### 2.11.1.21 cmd_mesh_config_client_get_request_status

Get the status of a pending request.

**Table 2.354. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x01 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |

**Table 2.355. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | server_address | Address of the Configuration Server |
| 8-9 | uint16 | opcode | Message opcode used in the request. Opcodes are defined in the Bluetooth mesh stack 1.0 specification. |
| 10-13 | uint32 | age | Time in milliseconds that the request has been pending |
| 14-17 | uint32 | remaining | Time in milliseconds before the request times out. Note that time-out may be adjusted if it's determined that the request is destined to an LPN, which may respond very slowly. |
| 18 | uint8 | friend_acked | If non-zero, response has been acknowledged by a Friend node, so it is destined to an LPN and may take a long time to complete. |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_get_request_status_rsp_t
*gecko_cmd_mesh_config_client_get_request_status(uint32 handle);

/* Response id */
gecko_rsp_mesh_config_client_get_request_status_id

/* Response structure */
struct gecko_msg_mesh_config_client_get_request_status_rsp_t
{
  uint16 result;,
  uint16 server_address;,
  uint16 opcode;,
  uint32 age;,
  uint32 remaining;,
```

```
  uint8 friend_acked;
};
```

#### 2.11.1.22 cmd_mesh_config_client_list_appkeys

List the application keys on a node.

**Table 2.356. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | netkey_index | Network key index for the key used as the query parameter. The result contains the indices of the application keys bound to this network key on the node. |

**Table 2.357. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_config_client_list_appkeys_rsp_t   *gecko_cmd_mesh_config_client_list_appkeys(uint16
enc_netkey_index, uint16 server_address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_config_client_list_appkeys_id

/* Response structure */
struct gecko_msg_mesh_config_client_list_appkeys_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.358. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_appkey_list | This event contains a list of key indices for application keys that are present on a node and are bound to the network key specified in the request. The list is requested using the application key list command. More than one event may be generated. List contents are terminated by a application key list end event. |
| mesh_config_client_appkey_list_end | Terminating event for application key index list |

#### 2.11.1.23 cmd_mesh_config_client_list_bindings

List application key bindings of a model.

**Table 2.359. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be queried resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to query. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to query |

**Table 2.360. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_list_bindings_rsp_t  *gecko_cmd_mesh_config_client_list_bindings(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_config_client_list_bindings_id

/* Response structure */
struct gecko_msg_mesh_config_client_list_bindings_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.361. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_bindings_list | This event contains a list of key indices for the application keys which are bound to a model. The list is requested using the list model-application key bindings command. More than one such event may be generated; the list contents are terminated by a model-application key bindings list end event. |
| mesh_config_client_bindings_list_end | Terminating event for model-application key bindings list |

#### 2.11.1.24 cmd_mesh_config_client_list_netkeys

List the network keys on a node.

**Table 2.362. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.363. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_config_client_list_netkeys_rsp_t   *gecko_cmd_mesh_config_client_list_netkeys(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_list_netkeys_id

/* Response structure */
struct gecko_msg_mesh_config_client_list_netkeys_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.364. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_netkey_list | This event contains a list of key indices for network keys that are present on a node. The list is requested using the network key list command. More than one event may be generated. List contents are terminated by a network key list end event. |
| mesh_config_client_netkey_list_end | Terminating event for network key index list |

#### 2.11.1.25 cmd_mesh_config_client_list_subs

Get the subscription address list of a model.

**Table 2.365. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be queried resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to query. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to query |

**Table 2.366. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct     gecko_msg_mesh_config_client_list_subs_rsp_t     *gecko_cmd_mesh_config_client_list_subs(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_config_client_list_subs_id

/* Response structure */
struct gecko_msg_mesh_config_client_list_subs_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.367. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_subs_list | This event contains a list of addresses the queried model subscribes to. The list is requested using the list subscription addresses command. More than one event may be generated. List contents are terminated by a subscription address list end event. Note that if the subscription address list entry is a Label UUID (full virtual address), the corresponding virtual address hash is returned in this event. |
| mesh_config_client_subs_list_end | Terminating event for model subscription list |

#### 2.11.1.26 cmd_mesh_config_client_remove_appkey

Remove an application key from a node.

**Table 2.368. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | appkey_index | Index of the application key to remove |
| 10-11 | uint16 | netkey_index | Index of the network key bound to the application key to on the node. Note that this may be different from the binding on other nodes or on the Configuration Client. |

**Table 2.369. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_remove_appkey_rsp_t  *gecko_cmd_mesh_config_client_remove_appkey(uint16
enc_netkey_index, uint16 server_address, uint16 appkey_index, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_config_client_remove_appkey_id

/* Response structure */
struct gecko_msg_mesh_config_client_remove_appkey_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.370. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_appkey_status | This event is created when a response for an add application key or a remove application key request is received or the request times out. |

#### 2.11.1.27 cmd_mesh_config_client_remove_model_sub

Remove an address from the model subscription list.

**Table 2.371. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-14 | uint16 | sub_address | The address to remove from the subscription list |

**Table 2.372. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_remove_model_sub_rsp_t
*gecko_cmd_mesh_config_client_remove_model_sub(uint16   enc_netkey_index,   uint16   server_address,   uint8
element_index, uint16 vendor_id, uint16 model_id, uint16 sub_address);

/* Response id */
gecko_rsp_mesh_config_client_remove_model_sub_id

/* Response structure */
struct gecko_msg_mesh_config_client_remove_model_sub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.373. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.28 cmd_mesh_config_client_remove_model_sub_va

Remove a Label UUID (full virtual address) from model subscription list.

**Table 2.374. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x19 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-28 | uuid_128 | sub_address | The full virtual address to remove from the subscription list |

**Table 2.375. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_remove_model_sub_va_rsp_t
*gecko_cmd_mesh_config_client_remove_model_sub_va(uint16   enc_netkey_index,   uint16   server_address,   uint8
element_index, uint16 vendor_id, uint16 model_id, uuid_128 sub_address);

/* Response id */
gecko_rsp_mesh_config_client_remove_model_sub_va_id

/* Response structure */
struct gecko_msg_mesh_config_client_remove_model_sub_va_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.376. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.29 cmd_mesh_config_client_remove_netkey

Remove a network key from a node.

**Table 2.377. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | netkey_index | Index of the network key to remove |

**Table 2.378. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_remove_netkey_rsp_t  *gecko_cmd_mesh_config_client_remove_netkey(uint16
enc_netkey_index, uint16 server_address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_config_client_remove_netkey_id

/* Response structure */
struct gecko_msg_mesh_config_client_remove_netkey_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.379. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_netkey_status | This event is created when a response for an add network key or a remove network key request is received, or the request times out. |

#### 2.11.1.30 cmd_mesh_config_client_reset_node

Request a node to unprovision itself. Use when a node is removed from the network.

**Table 2.380. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2d | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |

**Table 2.381. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_reset_node_rsp_t    *gecko_cmd_mesh_config_client_reset_node(uint16
enc_netkey_index, uint16 server_address);

/* Response id */
gecko_rsp_mesh_config_client_reset_node_id

/* Response structure */
struct gecko_msg_mesh_config_client_reset_node_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.382. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_reset_status | Indicates a node has received a reset request. |

#### 2.11.1.31 cmd_mesh_config_client_set_beacon

Set node secure network beacon state.

**Table 2.383. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1c | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | value | Secure network beacon value to set. Valid values are:<br>• 0: Node is not broadcasting secure network beacons<br>• 1: Node is broadcasting secure network beacons |

**Table 2.384. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_set_beacon_rsp_t    *gecko_cmd_mesh_config_client_set_beacon(uint16
enc_netkey_index, uint16 server_address, uint8 value);

/* Response id */
gecko_rsp_mesh_config_client_set_beacon_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_beacon_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.385. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_beacon_status | Status event for get beacon state and set beacon state commands. |

#### 2.11.1.32 cmd_mesh_config_client_set_default_timeout

Set the default timeout for configuration client requests.

**Table 2.386. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2f | method | Message ID |
| 4-7 | uint32 | timeout_ms | Timeout in milliseconds. Default timeout is 5 s (5000 ms). |
| 8-11 | uint32 | lpn_timeout_ms | Timeout in milliseconds when communicating with an LPN node. Default LPN timeout is 120 s (120000 ms). |

**Table 2.387. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x2f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_set_default_timeout_rsp_t
*gecko_cmd_mesh_config_client_set_default_timeout(uint32 timeout_ms, uint32 lpn_timeout_ms);

/* Response id */
gecko_rsp_mesh_config_client_set_default_timeout_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_default_timeout_rsp_t
{
  uint16 result;
};
```

#### 2.11.1.33 cmd_mesh_config_client_set_default_ttl

Set node default TTL state.

**Table 2.388. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1e | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | value | Default TTL value. Valid value range is from 2 to 127 for relayed PDUs, and 0 to indicate non-relayed PDUs |

**Table 2.389. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_config_client_set_default_ttl_rsp_t *gecko_cmd_mesh_config_client_set_default_ttl(uint16
enc_netkey_index, uint16 server_address, uint8 value);

/* Response id */
gecko_rsp_mesh_config_client_set_default_ttl_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_default_ttl_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.390. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_default_ttl_status | Status event for get default TTL state and set default TTL state commands. |

#### 2.11.1.34 cmd_mesh_config_client_set_friend

Set node friend state.

**Table 2.391. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x28 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | value | Friend value to set. Valid values are:<br>• 0: Friend feature is not enabled<br>• 1: Friend feature is enabled |

**Table 2.392. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x28 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_config_client_set_friend_rsp_t    *gecko_cmd_mesh_config_client_set_friend(uint16
enc_netkey_index, uint16 server_address, uint8 value);

/* Response id */
gecko_rsp_mesh_config_client_set_friend_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_friend_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.393. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_friend_status | Status event for get friend state and set friend state commands. |

#### 2.11.1.35 cmd_mesh_config_client_set_gatt_proxy

Set node GATT proxy state.

**Table 2.394. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x20 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | value | GATT proxy value to set. Valid values are:<br>• 0: Proxy feature is disabled<br>• 1: Proxy feature is enabled |

**Table 2.395. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x20 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_set_gatt_proxy_rsp_t  *gecko_cmd_mesh_config_client_set_gatt_proxy(uint16
enc_netkey_index, uint16 server_address, uint8 value);

/* Response id */
gecko_rsp_mesh_config_client_set_gatt_proxy_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_gatt_proxy_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.396. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_gatt_proxy_status | Status event for get GATT proxy state and set GATT proxy state commands |

#### 2.11.1.36 cmd_mesh_config_client_set_heartbeat_pub

Set the heartbeat publication state of a node.

**Table 2.397. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | destination_address | Heartbeat publication destination address. The address can't be a virtual address. Note that it can be the unassigned address, in which case the heartbeat publishing is disabled. |
| 10-11 | uint16 | netkey_index | Index of the network key used to encrypt heartbeat messages |
| 12 | uint8 | count_log | Heartbeat publication count logarithm-of-2 setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node will send 2^(n-1) heartbeat messages<br>• **0x12 .. 0xfe:** Prohibited<br>• **0xff:** Hearbeat messages are sent indefinitely |
| 13 | uint8 | period_log | Heartbeat publication period logarithm-of-2 setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node will send a heartbeat message every 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |
| 14 | uint8 | ttl | Time-to-live value for heartbeat messages |
| 15-16 | uint16 | features | Heartbeat trigger setting. For bits set in the bitmask, reconfiguration of the node feature associated with the bit will result in the node emitting a heartbeat message. Valid values are as follows:<br>• **Bit 0:** Relay feature<br>• **Bit 1:** Proxy feature<br>• **Bit 2:** Friend feature<br>• **Bit 3:** Low power feature<br>Remaining bits are reserved for future use. |

**Table 2.398. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x17 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_set_heartbeat_pub_rsp_t
*gecko_cmd_mesh_config_client_set_heartbeat_pub(uint16   enc_netkey_index,   uint16   server_address,   uint16
destination_address, uint16 netkey_index, uint8 count_log, uint8 period_log, uint8 ttl, uint16 features);

/* Response id */
gecko_rsp_mesh_config_client_set_heartbeat_pub_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_heartbeat_pub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.399. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_heartbeat_pub_status | Status event for get heartbeat publication state and set heartbeat publication state commands |

#### 2.11.1.37 cmd_mesh_config_client_set_heartbeat_sub

Set the heartbeat subscription state of a node.

**Table 2.400. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | source_address | Source address for heartbeat messages, which must be either a unicast address or the unassigned address, in which case heartbeat messages are not processed. |
| 10-11 | uint16 | destination_address | Destination address for heartbeat messages. The address must be either the unicast address of the primary element of the node, a group address, or the unassigned address. If it is the unassigned address, heartbeat messages are not processed. |
| 12 | uint8 | period_log | Heartbeat subscription period logarithm-of-2 setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not received<br>• **0x01 .. 0x11:** Node will receive heartbeat messages for 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |

**Table 2.401. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_set_heartbeat_sub_rsp_t
*gecko_cmd_mesh_config_client_set_heartbeat_sub(uint16   enc_netkey_index,   uint16   server_address,   uint16
source_address, uint16 destination_address, uint8 period_log);

/* Response id */
gecko_rsp_mesh_config_client_set_heartbeat_sub_id

/* Response structure */
```

```
struct gecko_msg_mesh_config_client_set_heartbeat_sub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.402. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_heartbeat_sub_status | Status event for get heartbeat subscription state and set heartbeat subscription state commands |

#### 2.11.1.38 cmd_mesh_config_client_set_identity

Set node identity state.

**Table 2.403. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x26 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8-9 | uint16 | netkey_index | Network key index for which the state is configured |
| 10 | uint8 | value | Identity value to set. Valid values are:<br>• 0: Node identity advertising is disabled<br>• 1: Node identity advertising is enabled |

**Table 2.404. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x26 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_config_client_set_identity_rsp_t   *gecko_cmd_mesh_config_client_set_identity(uint16
enc_netkey_index, uint16 server_address, uint16 netkey_index, uint8 value);

/* Response id */
gecko_rsp_mesh_config_client_set_identity_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_identity_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.405. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_identity_status | Status event for get node identity state and set node identity state commands. |

#### 2.11.1.39 cmd_mesh_config_client_set_model_pub

Set model publication state.

**Table 2.406. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x16 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-14 | uint16 | address | The address to publish to. Can be a unicast address, a virtual address, or a group address; can also be the unassigned address to stop the model from publishing. |
| 15-16 | uint16 | appkey_index | The application key index to use for the published messages. |
| 17 | uint8 | credentials | Friendship credential flag. If zero, publication is done using normal credentials. If one, it is done with friendship credentials, meaning only the friend can decrypt the published message and relay it forward using the normal credentials. The default value is 0. |
| 18 | uint8 | ttl | Publication time-to-live value |
| 19-22 | uint32 | period_ms | Publication period in milliseconds. Note that the resolution of the publication period is limited by the specification to 100 ms up to a period of 6.3 s, 1 s up to a period of 63 s, 10 s up to a period of 630 s, and 10 minutes above that. Maximum period allowed is 630 minutes. |
| 23 | uint8 | retransmit_count | Publication retransmission count. Valid values range from 0 to 7. |
| 24-25 | uint16 | retransmit_interval_ms | Publication retransmission interval in millisecond units. The range of value is 50 to 1600 ms, and the resolution of the value is 50 milliseconds. |

**Table 2.407. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0c | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_set_model_pub_rsp_t  *gecko_cmd_mesh_config_client_set_model_pub(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 vendor_id, uint16 model_id, uint16
address, uint16 appkey_index, uint8 credentials, uint8 ttl, uint32 period_ms, uint8 retransmit_count, uint16
retransmit_interval_ms);

/* Response id */
gecko_rsp_mesh_config_client_set_model_pub_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_model_pub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.408. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_pub_status | Status event for get model publication state, set model publication state, commands. |

#### 2.11.1.40 cmd_mesh_config_client_set_model_pub_va

Set model publication state, with a full virtual publication address.

**Table 2.409. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x24 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-28 | uuid_128 | address | The Label UUID (full virtual address) to publish to |
| 29-30 | uint16 | appkey_index | The application key index to use for the published messages |
| 31 | uint8 | credentials | Friendship credential flag. If zero, publication is done using normal credentials. If one, it is done with friendship credentials, meaning only the friend can decrypt the published message and relay it forward using the normal credentials. The default value is 0. |
| 32 | uint8 | ttl | Publication time-to-live value |
| 33-36 | uint32 | period_ms | Publication period in milliseconds. Note that the resolution of the publication period is limited by the specification to 100 ms up to a period of 6.3 s, 1 s up to a period of 63 s, 10 s up to a period of 630 s, and 10 minutes above that. Maximum period allowed is 630 minutes. |
| 37 | uint8 | retransmit_count | Publication retransmission count. Valid values range from 0 to 7. |
| 38-39 | uint16 | retransmit_interval_ms | Publication retransmission interval in millisecond units. The range of value is 50 to 1600 ms. The resolution of the value is 50 milliseconds. |

**Table 2.410. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

| Byte | Type | Name | Description |
|---|---|---|---|
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_set_model_pub_va_rsp_t
*gecko_cmd_mesh_config_client_set_model_pub_va(uint16   enc_netkey_index,   uint16   server_address,   uint8
element_index, uint16 vendor_id, uint16 model_id, uuid_128 address, uint16 appkey_index, uint8 credentials,
uint8 ttl, uint32 period_ms, uint8 retransmit_count, uint16 retransmit_interval_ms);

/* Response id */
gecko_rsp_mesh_config_client_set_model_pub_va_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_model_pub_va_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.411. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_pub_status | Status event for get model publication state, set model publication state, commands. |

#### 2.11.1.41 cmd_mesh_config_client_set_model_sub

Set (overwrite) model subscription address list to a single address.

**Table 2.412. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-14 | uint16 | sub_address | The address to set as the subscription list |

**Table 2.413. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_config_client_set_model_sub_rsp_t  *gecko_cmd_mesh_config_client_set_model_sub(uint16
enc_netkey_index,  uint16  server_address,  uint8  element_index,  uint16  vendor_id,  uint16  model_id,  uint16
sub_address);

/* Response id */
gecko_rsp_mesh_config_client_set_model_sub_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_model_sub_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.414. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.42 cmd_mesh_config_client_set_model_sub_va

Set (overwrite) model subscription address list to a single virtual address.

**Table 2.415. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x19 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 11-12 | uint16 | model_id | Model ID for the model to configure |
| 13-28 | uuid_128 | sub_address | The full virtual address to set as the subscription list |

**Table 2.416. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_set_model_sub_va_rsp_t
*gecko_cmd_mesh_config_client_set_model_sub_va(uint16   enc_netkey_index,   uint16   server_address,   uint8
element_index, uint16 vendor_id, uint16 model_id, uuid_128 sub_address);

/* Response id */
gecko_rsp_mesh_config_client_set_model_sub_va_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_model_sub_va_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.417. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_model_sub_status | Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands |

#### 2.11.1.43 cmd_mesh_config_client_set_network_transmit

Set node network transmit state.

**Table 2.418. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x24 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | transmit_count | Network transmit count. Valid values range from 1 to 8; default value is 1 (single transmission; no retransmissions). |
| 9-10 | uint16 | transmit_interval_ms | Network transmit interval in milliseconds. Valid values range from 10 ms to 320 ms, with a resolution of 10 ms. Value is ignored if the transmission count is set to one. |

**Table 2.419. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x24 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_config_client_set_network_transmit_rsp_t
*gecko_cmd_mesh_config_client_set_network_transmit(uint16  enc_netkey_index,  uint16  server_address,  uint8
transmit_count, uint16 transmit_interval_ms);

/* Response id */
gecko_rsp_mesh_config_client_set_network_transmit_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_network_transmit_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.420. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_network_transmit_status | Status event for get network transmit state and set network transmit state commands |

#### 2.11.1.44 cmd_mesh_config_client_set_relay

Set node relay state.

**Table 2.421. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x22 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | value | Relay value to set. Valid values are:<br>• 0: Relay feature is disabled<br>• 1: Relay feature is enabled |
| 9 | uint8 | retransmit_count | Relay retransmit count. Valid values range from 0 to 7; default value is 0 (no retransmissions). |
| 10-11 | uint16 | retransmit_interval_ms | Relay retransmit interval in milliseconds. Valid values range from 10 ms to 320 ms, with a resolution of 10 ms. Value is ignored if retransmission count is set to zero. |

**Table 2.422. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x22 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct     gecko_msg_mesh_config_client_set_relay_rsp_t     *gecko_cmd_mesh_config_client_set_relay(uint16
enc_netkey_index, uint16 server_address, uint8 value, uint8 retransmit_count, uint16 retransmit_interval_ms);

/* Response id */
gecko_rsp_mesh_config_client_set_relay_id

/* Response structure */
struct gecko_msg_mesh_config_client_set_relay_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.423. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_relay_status | Status event for get relay state and set relay state commands |

#### 2.11.1.45 cmd_mesh_config_client_unbind_model

Unbind an application key from a model

**Table 2.424. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | enc_netkey_index | Network key used to encrypt the request on the network layer |
| 6-7 | uint16 | server_address | Destination node primary element address |
| 8 | uint8 | element_index | Index of the element where the model to be configured resides on the node |
| 9-10 | uint16 | appkey_index | Index of the application key to unbind from the model |
| 11-12 | uint16 | vendor_id | Vendor ID for the model to configure. Use 0xFFFF for Bluetooth SIG models. |
| 13-14 | uint16 | model_id | Model ID for the model to configure |

**Table 2.425. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_config_client_unbind_model_rsp_t   *gecko_cmd_mesh_config_client_unbind_model(uint16
enc_netkey_index, uint16 server_address, uint8 element_index, uint16 appkey_index, uint16 vendor_id, uint16
model_id);

/* Response id */
gecko_rsp_mesh_config_client_unbind_model_id

/* Response structure */
struct gecko_msg_mesh_config_client_unbind_model_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

**Table 2.426. Events Generated**

| Event | Description |
|---|---|
| mesh_config_client_binding_status | Status event for binding and unbinding application keys and models. |

### 2.11.2 mesh_config_client events

#### 2.11.2.1 evt_mesh_config_client_appkey_list

This event contains a list of key indices for application keys that are present on a node and are bound to the network key specified in the request. The list is requested using the application key list command. More than one event may be generated. List contents are terminated by a application key list end event.

**Table 2.427. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x05 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |
| 8 | uint8array | appkey_indices | List of application key indices, two bytes per entry |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_appkey_list_id

/* Event structure */
struct gecko_msg_mesh_config_client_appkey_list_evt_t
{
  uint32 handle;,
  uint8array appkey_indices;
};
```

#### 2.11.2.2 evt_mesh_config_client_appkey_list_end

Terminating event for application key index list

**Table 2.428. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_appkey_list_end_id

/* Event structure */
struct gecko_msg_mesh_config_client_appkey_list_end_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.3 evt_mesh_config_client_appkey_status

This event is created when a response for an add application key or a remove application key request is received or the request times out.

**Table 2.429. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_appkey_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_appkey_status_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.4 evt_mesh_config_client_beacon_status

Status event for get beacon state and set beacon state commands.

**Table 2.430. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | value | Secure network beacon state of the node. Valid values are:<br>• 0: Node is not broadcasting secure network beacons<br>• 1: Node is broadcasting secure network beacons |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_beacon_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_beacon_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 value;
};
```

#### 2.11.2.5 evt_mesh_config_client_binding_status

Status event for binding and unbinding application keys and models.

**Table 2.431. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_binding_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_binding_status_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.6 evt_mesh_config_client_bindings_list

This event contains a list of key indices for the application keys which are bound to a model. The list is requested using the list model-application key bindings command. More than one such event may be generated; the list contents are terminated by a model-application key bindings list end event.

**Table 2.432. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x08 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |
| 8 | uint8array | appkey_indices | List of application key indices, two bytes per entry |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_bindings_list_id

/* Event structure */
struct gecko_msg_mesh_config_client_bindings_list_evt_t
{
  uint32 handle;,
  uint8array appkey_indices;
};
```

#### 2.11.2.7 evt_mesh_config_client_bindings_list_end

Terminating event for model-application key bindings list

**Table 2.433. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_bindings_list_end_id

/* Event structure */
struct gecko_msg_mesh_config_client_bindings_list_end_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.8 evt_mesh_config_client_dcd_data

Event reporting queried composition data page contents. The contents are requested using the get device composition data command. More than one event may be generated. Page contents are terminated by a composition data end event. Note that the interpretation of the received data is page-specific. Page 0 contains the element and model layout of the node.

**Table 2.434. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x19 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |
| 8 | uint8 | page | Composition data page containing data |
| 9 | uint8array | data | Composition data page contents |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_dcd_data_id

/* Event structure */
struct gecko_msg_mesh_config_client_dcd_data_evt_t
{
  uint32 handle;,
  uint8 page;,
  uint8array data;
};
```

#### 2.11.2.9 evt_mesh_config_client_dcd_data_end

Terminating event for node composition data

**Table 2.435. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_dcd_data_end_id

/* Event structure */
struct gecko_msg_mesh_config_client_dcd_data_end_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.10 evt_mesh_config_client_default_ttl_status

Status event for get default TTL state and set default TTL state commands.

**Table 2.436. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | value | Default TTL value. Valid value range is from 2 to 127 for relayed PDUs, and 0 to indicate non-relayed PDUs. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_default_ttl_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_default_ttl_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 value;
};
```

#### 2.11.2.11 evt_mesh_config_client_friend_status

Status event for get friend state and set friend state commands.

**Table 2.437. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | value | Friend value to set. Valid values are:<br>• 0: Friend feature is not enabled<br>• 1: Friend feature is enabled<br>• 2: Friend feature is not supported |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_friend_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_friend_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 value;
};
```

#### 2.11.2.12 evt_mesh_config_client_gatt_proxy_status

Status event for get GATT proxy state and set GATT proxy state commands

**Table 2.438. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | value | GATT proxy value of the node. Valid values are:<br>• 0: GATT proxy feature is disabled<br>• 1: GATT proxy feature is enabled<br>• 2: GATT proxy feature is not supported |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_gatt_proxy_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_gatt_proxy_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 value;
};
```

#### 2.11.2.13 evt_mesh_config_client_heartbeat_pub_status

Status event for get heartbeat publication state and set heartbeat publication state commands

**Table 2.439. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10-11 | uint16 | destination_address | Heartbeat publication destination address. |
| 12-13 | uint16 | netkey_index | Index of the network key used to encrypt heartbeat messages |
| 14 | uint8 | count_log | Heartbeat publication count logarithm-of-2 setting |
| 15 | uint8 | period_log | Heartbeat publication period logarithm-of-2 setting |
| 16 | uint8 | ttl | Time-to-live value for heartbeat messages |
| 17-18 | uint16 | features | Heartbeat trigger setting |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_heartbeat_pub_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_heartbeat_pub_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint16 destination_address;,
  uint16 netkey_index;,
  uint8 count_log;,
  uint8 period_log;,
  uint8 ttl;,
  uint16 features;
};
```

#### 2.11.2.14 evt_mesh_config_client_heartbeat_sub_status

Status event for get heartbeat subscription state and set heartbeat subscription state commands

**Table 2.440. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10-11 | uint16 | source_address | Source address for heartbeat messages |
| 12-13 | uint16 | destination_address | Destination address for heartbeat messages |
| 14 | uint8 | period_log | Heartbeat subscription remaining period logarithm-of-2 value |
| 15 | uint8 | count_log | Received heartbeat message count logarithm-of-2 value |
| 16 | uint8 | min_hops | Minimum hop value seen in received heartbeat messages |
| 17 | uint8 | max_hops | Minimum hop value seen in received heartbeat messages |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_heartbeat_sub_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_heartbeat_sub_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint16 source_address;,
  uint16 destination_address;,
  uint8 period_log;,
  uint8 count_log;,
  uint8 min_hops;,
  uint8 max_hops;
};
```

#### 2.11.2.15 evt_mesh_config_client_identity_status

Status event for get node identity state and set node identity state commands.

**Table 2.441. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | value | Identity state of the node for the used network index. Valid values are as follows:<br>• 0: Node identity advertising is disabled<br>• 1: Node identity advertising is enabled<br>• 2: Node identity advertising is not supported |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_identity_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_identity_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 value;
};
```

#### 2.11.2.16 evt_mesh_config_client_lpn_polltimeout_status

Status event for get LPN poll timeout command.

**Table 2.442. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10-13 | uint32 | poll_timeout_ms | Poll timeout value, in milliseconds, for the specified LPN. The value reported is zero if the queried Friend does not have an ongoing friendship with the specified LPN. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_lpn_polltimeout_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_lpn_polltimeout_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint32 poll_timeout_ms;
};
```

#### 2.11.2.17 evt_mesh_config_client_model_pub_status

Status event for get model publication state, set model publication state, commands.

**Table 2.443. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x13 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10-11 | uint16 | address | The address to publish to. If this address is the unassigned address, the model is prevented from publishing. Note that if state contains a Label UUID (full virtual address), the corresponding virtual address hash is returned in this parameter. |
| 12-13 | uint16 | appkey_index | The application key index used for the published messages |
| 14 | uint8 | credentials | Friendship credentials flag |
| 15 | uint8 | ttl | Publication time-to-live value |
| 16-19 | uint32 | period_ms | Publication period in milliseconds |
| 20 | uint8 | retransmit_count | Publication retransmission count |
| 21-22 | uint16 | retransmit_interval_ms | Publication retransmission interval in milliseconds |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_model_pub_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_model_pub_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint16 address;,
  uint16 appkey_index;,
  uint8 credentials;,
  uint8 ttl;,
  uint32 period_ms;,
  uint8 retransmit_count;,
  uint16 retransmit_interval_ms;
};
```

#### 2.11.2.18 evt_mesh_config_client_model_sub_status

Status event for add subscription address, , remove subscription address, , set subscription address, , and clear subscription address list commands

**Table 2.444. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_model_sub_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_model_sub_status_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.19 evt_mesh_config_client_netkey_list

This event contains a list of key indices for network keys that are present on a node. The list is requested using the network key list command. More than one event may be generated. List contents are terminated by a network key list end event.

**Table 2.445. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x02 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |
| 8 | uint8array | netkey_indices | List of network key indices, two bytes per entry |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_netkey_list_id

/* Event structure */
struct gecko_msg_mesh_config_client_netkey_list_evt_t
{
  uint32 handle;,
  uint8array netkey_indices;
};
```

#### 2.11.2.20 evt_mesh_config_client_netkey_list_end

Terminating event for network key index list

**Table 2.446. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_netkey_list_end_id

/* Event structure */
struct gecko_msg_mesh_config_client_netkey_list_end_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.21 evt_mesh_config_client_netkey_status

This event is created when a response for an add network key or a remove network key request is received, or the request times out.

**Table 2.447. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_netkey_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_netkey_status_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.22 evt_mesh_config_client_network_transmit_status

Status event for get network transmit state and set network transmit state commands

**Table 2.448. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | transmit_count | Network transmit count. Valid values range from 1 to 8; default value is 1 (single transmission; no retransmissions). |
| 11-12 | uint16 | transmit_interval_ms | Network transmit interval in milliseconds. Valid values range from 10 ms to 320 ms, with a resolution of 10 ms. Value will be zero if the transmission count is set to one. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_network_transmit_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_network_transmit_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 transmit_count;,
  uint16 transmit_interval_ms;
};
```

#### 2.11.2.23 evt_mesh_config_client_relay_status

Status event for get relay state and set relay state commands

**Table 2.449. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |
| 10 | uint8 | relay | Relay state of the node. Valid values are as follows:<br>• 0: Relaying disabled<br>• 1: Relaying enabled<br>• 2: Relaying not supported |
| 11 | uint8 | retransmit_count | Relay retransmit count. Valid values range from 0 to 7; default value is 0 (no retransmissions). |
| 12-13 | uint16 | retransmit_interval_ms | Relay retransmit interval in milliseconds. Valid values range from 10 ms to 320 ms, with a resolution of 10 ms. Value will be zero if retransmission count is zero. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_relay_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_relay_status_evt_t
{
  uint16 result;,
  uint32 handle;,
  uint8 relay;,
  uint8 retransmit_count;,
  uint16 retransmit_interval_ms;
};
```

#### 2.11.2.24 evt_mesh_config_client_request_modified

Pending request parameters have been updated. The application may call get request status command to retrieve the current status of the request. This event is generated when the timeout of a request is extended because the request is acknowledged by a Friend node on behalf of the LPN, which is the destination of the request.

**Table 2.450. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_request_modified_id

/* Event structure */
struct gecko_msg_mesh_config_client_request_modified_evt_t
{
  uint32 handle;
};
```

#### 2.11.2.25 evt_mesh_config_client_reset_status

Indicates a node has received a reset request.

**Table 2.451. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_reset_status_id

/* Event structure */
struct gecko_msg_mesh_config_client_reset_status_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.11.2.26 evt_mesh_config_client_subs_list

This event contains a list of addresses the queried model subscribes to. The list is requested using the list subscription addresses command. More than one event may be generated. List contents are terminated by a subscription address list end event. Note that if the subscription address list entry is a Label UUID (full virtual address), the corresponding virtual address hash is returned in this event.

**Table 2.452. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0c | method | Message ID |
| 4-7 | uint32 | handle | Request handle |
| 8 | uint8array | addresses | List of subscription addresses, two bytes per entry |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_subs_list_id

/* Event structure */
struct gecko_msg_mesh_config_client_subs_list_evt_t
{
  uint32 handle;,
  uint8array addresses;
};
```

#### 2.11.2.27 evt_mesh_config_client_subs_list_end

Terminating event for model subscription list

**Table 2.453. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x27 | class | Message class: Bluetooth Mesh Configuration Client |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | Request handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_config_client_subs_list_end_id

/* Event structure */
struct gecko_msg_mesh_config_client_subs_list_end_evt_t
{
  uint16 result;,
  uint32 handle;
};
```

## 2.12 Bluetooth Mesh Friend Node API (mesh_friend)

These commands and events are for Friend operation, available in nodes which have the Friend feature.

### 2.12.1 mesh_friend commands

#### 2.12.1.1 cmd_mesh_friend_deinit

Deinitialize the Friend functionality. After calling this command, a possible friendship with a Low Power node is terminated and all friendships are terminated. After calling this command, don't call other commands in this class before the Friend mode is initialized again.

**Table 2.454. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x24 | class | Message class: Bluetooth Mesh Friend Node API |
| 3 | 0x01 | method | Message ID |

**Table 2.455. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x24 | class | Message class: Bluetooth Mesh Friend Node API |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_friend_deinit_rsp_t *gecko_cmd_mesh_friend_deinit();

/* Response id */
gecko_rsp_mesh_friend_deinit_id

/* Response structure */
struct gecko_msg_mesh_friend_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.12.1.2 cmd_mesh_friend_init

Initialize the Friend mode. The node needs to be provisioned before calling this command. After the Friend mode is initialized, it is ready to accept friend requests from low-power nodes. This call has to be made before calling the other commands in this class.

**Table 2.456. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x24 | class | Message class: Bluetooth Mesh Friend Node API |
| 3 | 0x00 | method | Message ID |

**Table 2.457. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x24 | class | Message class: Bluetooth Mesh Friend Node API |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_friend_init_rsp_t *gecko_cmd_mesh_friend_init();

/* Response id */
gecko_rsp_mesh_friend_init_id

/* Response structure */
struct gecko_msg_mesh_friend_init_rsp_t
{
  uint16 result;
};
```

### 2.12.2 mesh_friend events

#### 2.12.2.1 evt_mesh_friend_friendship_established

Indication that a friendship has been established

**Table 2.458. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x24 | class | Message class: Bluetooth Mesh Friend Node API |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | lpn_address | LPN node address |
| 6-7 | uint16 | netkey_index | Index of the network key used in friendship |

**C Functions**

```
/* Event id */
gecko_evt_mesh_friend_friendship_established_id

/* Event structure */
struct gecko_msg_mesh_friend_friendship_established_evt_t
{
  uint16 lpn_address;,
  uint16 netkey_index;
};
```

#### 2.12.2.2 evt_mesh_friend_friendship_terminated

Indication that a friendship that was successfully established has been terminated

**Table 2.459. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x24 | class | Message class: Bluetooth Mesh Friend Node API |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | reason | Reason for friendship termination |
| 6-7 | uint16 | lpn_address | LPN node address |
| 8-9 | uint16 | netkey_index | Index of the network key used in friendship |

**C Functions**

```
/* Event id */
gecko_evt_mesh_friend_friendship_terminated_id

/* Event structure */
struct gecko_msg_mesh_friend_friendship_terminated_evt_t
{
  uint16 reason;,
  uint16 lpn_address;,
  uint16 netkey_index;
};
```

## 2.13 Bluetooth Mesh Generic Client Model (mesh_generic_client)

Generic client model API provides functionality to send and receive messages using Bluetooth SIG client models, including generic client models and lighting client models.

Throughout the API, the client model being used is identified by its element address and model ID, while the server model responding to client model requests is identified by its element address and model ID.

The API has functions for querying server model states, requesting server model state changes, and publishing messages. The application has to implement more complex functionality (state machines or other model-specific logic).

Data for state change requests and server responses is passed as serialized byte arrays through BGAPI. There are functions to convert byte arrays to and from model state structures in the Bluetooth mesh SDK.

The stack will handle Mesh transaction layer segmentation and reassembly automatically if the messages sent are long enough to require it.

**Note on time resolution**

Because of message formatting, transition time and remaining time resolution units depend on the requested or reported value. For example, until 6.2 seconds it is 100 ms; until 62 seconds it is 1 s; until 620 seconds it is 10 s; and until 620 minutes it is 10 minutes. The value cannot be longer than 620 minutes. Therefore, it is not possible to request a delay of exactly 7500 ms. The resolution unit is 1 s between 6.2 and 62 seconds, so the value would be rounded down to 7 s.

Delay resolution is 5 ms and values will be rounded down to the closest 5 ms. The value can't be longer than 1275 ms.

### 2.13.1 mesh_generic_client commands

#### 2.13.1.1 cmd_mesh_generic_client_get

Get the current state of a server model or models in the network. Besides the immediate result code, the response or responses from the network will generate server state report events for the replies received.

The server model responses will be reported in server status events.

**Table 2.460. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | model_id | Client model ID |
| 6-7 | uint16 | elem_index | Client model element index |
| 8-9 | uint16 | server_address | Destination server model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | type | Model-specific state type, identifying the kind of state to retrieve. See get state types list for details. |

**Table 2.461. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_generic_client_get_rsp_t  *gecko_cmd_mesh_generic_client_get(uint16  model_id,  uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 type);

/* Response id */
gecko_rsp_mesh_generic_client_get_id

/* Response structure */
struct gecko_msg_mesh_generic_client_get_rsp_t
{
  uint16 result;
};
```

**Table 2.462. Events Generated**

| Event | Description |
|---|---|
| mesh_generic_client_server_status | Status report sent by a server model. This may be generated either because of a response to a get or set request was received by the client model or because the client model received a spontaneously generated status indication sent to an address the model was subscribed to. |

### 2.13.1.2 cmd_mesh_generic_client_get_params

Get the current state of a server model or models in the network, with additional parameters detailing the request. Besides the immediate result code, the response or responses from the network will generate server state report events for the replies received.

The server model responses will be reported in server status events.

This call is used to query properties, for which the property ID is given as a parameter.

**Table 2.463. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | model_id | Client model ID |
| 6-7 | uint16 | elem_index | Client model element index |
| 8-9 | uint16 | server_address | Destination server model address |
| 10-11 | uint16 | appkey_index | The application key index to use. |
| 12 | uint8 | type | Model-specific state type, identifying the kind of state to retrieve. See get state types list for details. |
| 13 | uint8array | parameters | Message-specific get request parameters serialized into a byte array |

**Table 2.464. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_generic_client_get_params_rsp_t    *gecko_cmd_mesh_generic_client_get_params(uint16
model_id, uint16 elem_index, uint16 server_address, uint16 appkey_index, uint8 type, uint8 parameters_len,
const uint8 *parameters_data);

/* Response id */
gecko_rsp_mesh_generic_client_get_params_id

/* Response structure */
struct gecko_msg_mesh_generic_client_get_params_rsp_t
{
  uint16 result;
};
```

**Table 2.465. Events Generated**

| Event | Description |
|---|---|
| mesh_generic_client_server_status | Status report sent by a server model. This may be generated either because of a response to a get or set request was received by the client model or because the client model received a spontaneously generated status indication sent to an address the model was subscribed to. |

#### 2.13.1.3 cmd_mesh_generic_client_init

Initialize generic client models. This command initializes all generic client models on the device. Alternatively, only the necessary client models can be initialized using model-specific initialization commands. Using model-specific initialization can result in a smaller firmware image size for SoC projects.

**Table 2.466. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x04 | method | Message ID |

**Table 2.467. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_rsp_t *gecko_cmd_mesh_generic_client_init();

/* Response id */
gecko_rsp_mesh_generic_client_init_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.4 cmd_mesh_generic_client_init_battery

Initialize generic battery client models

**Table 2.468. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0b | method | Message ID |

**Table 2.469. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_battery_rsp_t *gecko_cmd_mesh_generic_client_init_battery();

/* Response id */
gecko_rsp_mesh_generic_client_init_battery_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_battery_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.5 cmd_mesh_generic_client_init_common

Initialize generic client model common functionality. This should be called after all model-specific initialization calls are done, and does not need to be called if command to initialize all generic client models is used.

**Table 2.470. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x05 | method | Message ID |

**Table 2.471. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_common_rsp_t *gecko_cmd_mesh_generic_client_init_common();

/* Response id */
gecko_rsp_mesh_generic_client_init_common_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_common_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.6 cmd_mesh_generic_client_init_ctl

Initialize light CTL client models

**Table 2.472. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0f | method | Message ID |

**Table 2.473. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_ctl_rsp_t *gecko_cmd_mesh_generic_client_init_ctl();

/* Response id */
gecko_rsp_mesh_generic_client_init_ctl_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_ctl_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.7 cmd_mesh_generic_client_init_default_transition_time

Initialize generic default transition time client models

**Table 2.474. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x08 | method | Message ID |

**Table 2.475. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                    gecko_msg_mesh_generic_client_init_default_transition_time_rsp_t
*gecko_cmd_mesh_generic_client_init_default_transition_time();

/* Response id */
gecko_rsp_mesh_generic_client_init_default_transition_time_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_default_transition_time_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.8 cmd_mesh_generic_client_init_hsl

Initialize light HSL client models

**Table 2.476. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x10 | method | Message ID |

**Table 2.477. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_hsl_rsp_t *gecko_cmd_mesh_generic_client_init_hsl();

/* Response id */
gecko_rsp_mesh_generic_client_init_hsl_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_hsl_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.9 cmd_mesh_generic_client_init_level

Initialize generic level client models

**Table 2.478. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x07 | method | Message ID |

**Table 2.479. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_level_rsp_t *gecko_cmd_mesh_generic_client_init_level();

/* Response id */
gecko_rsp_mesh_generic_client_init_level_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_level_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.10 cmd_mesh_generic_client_init_lightness

Initialize light lightness client models

**Table 2.480. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0e | method | Message ID |

**Table 2.481. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_lightness_rsp_t *gecko_cmd_mesh_generic_client_init_lightness();

/* Response id */
gecko_rsp_mesh_generic_client_init_lightness_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_lightness_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.11 cmd_mesh_generic_client_init_location

Initialize generic location client models

**Table 2.482. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0c | method | Message ID |

**Table 2.483. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_location_rsp_t *gecko_cmd_mesh_generic_client_init_location();

/* Response id */
gecko_rsp_mesh_generic_client_init_location_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_location_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.12 cmd_mesh_generic_client_init_on_off

Initialize generic on/off client models

**Table 2.484. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x06 | method | Message ID |

**Table 2.485. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_on_off_rsp_t *gecko_cmd_mesh_generic_client_init_on_off();

/* Response id */
gecko_rsp_mesh_generic_client_init_on_off_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_on_off_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.13 cmd_mesh_generic_client_init_power_level

Initialize generic power level client models

**Table 2.486. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0a | method | Message ID |

**Table 2.487. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_power_level_rsp_t *gecko_cmd_mesh_generic_client_init_power_level();

/* Response id */
gecko_rsp_mesh_generic_client_init_power_level_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_power_level_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.14 cmd_mesh_generic_client_init_power_on_off

Initialize generic power on/off client models

**Table 2.488. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x09 | method | Message ID |

**Table 2.489. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_generic_client_init_power_on_off_rsp_t
*gecko_cmd_mesh_generic_client_init_power_on_off();

/* Response id */
gecko_rsp_mesh_generic_client_init_power_on_off_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_power_on_off_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.15 cmd_mesh_generic_client_init_property

Initialize generic property client models

**Table 2.490. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0d | method | Message ID |

**Table 2.491. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_client_init_property_rsp_t *gecko_cmd_mesh_generic_client_init_property();

/* Response id */
gecko_rsp_mesh_generic_client_init_property_id

/* Response structure */
struct gecko_msg_mesh_generic_client_init_property_rsp_t
{
  uint16 result;
};
```

#### 2.13.1.16 cmd_mesh_generic_client_publish

Publish a set request to the network using the publish address and publish application key of the model. The message will be received by the server models which subscribe to the publish address, and there's no need to explicitly specify a destination address or application key.

The server model responses will be reported in server status events. Note that for responses to be generated the corresponding flag needs to be set.

**Table 2.492. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | model_id | Client model ID |
| 6-7 | uint16 | elem_index | Client model element index |
| 8 | uint8 | tid | Transaction identifier |
| 9-12 | uint32 | transition | Transition time (in milliseconds) for the state change. If both the transition time and the delay are zero the transition is immediate.<br>This applies to messages the Mesh Model specification defines to have transition and delay times and can be left as zero for others. |
| 13-14 | uint16 | delay | Delay time (in milliseconds) before starting the state change. If both the transition time and the delay are zero the transition is immediate.<br>This applies to messages the Mesh Model specification defines to have transition and delay times, and can be left as zero for others. |
| 15-16 | uint16 | flags | Message flags. Bitmask of the following:<br>• **Bit 0:** Response required. If non-zero client expects a response from the server<br>• **Bit 1:** Default transition timer. If non-zero client requests that server uses its default transition timer and the supplied transition and delay values are ignored. |
| 17 | uint8 | type | Model-specific request type. See set request types list for details. |
| 18 | uint8array | parameters | Message-specific set request parameters serialized into a byte array |

**Table 2.493. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x02 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_generic_client_publish_rsp_t  *gecko_cmd_mesh_generic_client_publish(uint16  model_id,
uint16 elem_index, uint8 tid, uint32 transition, uint16 delay, uint16 flags, uint8 type, uint8 parameters_len,
const uint8 *parameters_data);

/* Response id */
gecko_rsp_mesh_generic_client_publish_id

/* Response structure */
struct gecko_msg_mesh_generic_client_publish_rsp_t
{
  uint16 result;
};
```

**Table 2.494. Events Generated**

| Event | Description |
|---|---|
| mesh_generic_client_server_status | Status report sent by a server model. This may be generated either because of a response to a get or set request was received by the client model or because the client model received a spontaneously generated status indication sent to an address the model was subscribed to. |

#### 2.13.1.17 cmd_mesh_generic_client_set

Set the current state of a server model or models in the network. Besides the immediate result code, the response or responses from the network will generate server state report events for the replies received.

The server model responses will be reported in server status events. Note that for responses to be generated the corresponding flag needs to be set.

**Table 2.495. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x13 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | model_id | Client model ID |
| 6-7 | uint16 | elem_index | Client model element index |
| 8-9 | uint16 | server_address | Destination server model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | tid | Transaction identifier. This applies to those messages the Mesh Model specification defines as transactional and can be left as zero for others. |
| 13-16 | uint32 | transition | Transition time (in milliseconds) for the state change. If both the transition time and the delay are zero the transition is immediate.<br>This applies to messages the Mesh Model specification defines to have transition and delay times and can be left as zero for others. |
| 17-18 | uint16 | delay | Delay time (in milliseconds) before starting the state change. If both the transition time and the delay are zero the transition is immediate.<br>This applies to messages the Mesh Model specification defines to have transition and delay times and can be left as zero for others. |
| 19-20 | uint16 | flags | Message flags. Bitmask of the following:<br>• **Bit 0:** Response required. If non-zero client expects a response from the server<br>• **Bit 1:** Default transition timer. If non-zero client requests that server uses its default transition timer and the supplied transition and delay values are ignored. |
| 21 | uint8 | type | Model-specific request type. See set request types list for details. |
| 22 | uint8array | parameters | Message-specific set request parameters serialized into a byte array |

**Table 2.496. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x01 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_generic_client_set_rsp_t  *gecko_cmd_mesh_generic_client_set(uint16  model_id,  uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 tid, uint32 transition, uint16 delay, uint16
flags, uint8 type, uint8 parameters_len, const uint8 *parameters_data);

/* Response id */
gecko_rsp_mesh_generic_client_set_id

/* Response structure */
struct gecko_msg_mesh_generic_client_set_rsp_t
{
  uint16 result;
};
```

**Table 2.497. Events Generated**

| Event | Description |
|---|---|
| mesh_generic_client_server_status | Status report sent by a server model. This may be generated either because of a response to a get or set request was received by the client model or because the client model received a spontaneously generated status indication sent to an address the model was subscribed to. |

### 2.13.2 mesh_generic_client events

#### 2.13.2.1 evt_mesh_generic_client_server_status

Status report sent by a server model. This may be generated either because of a response to a get or set request was received by the client model or because the client model received a spontaneously generated status indication sent to an address the model was subscribed to.

**Table 2.498. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x1e | class | Message class: Bluetooth Mesh Generic Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | model_id | Client model ID |
| 6-7 | uint16 | elem_index | Client model element index |
| 8-9 | uint16 | client_address | Address that the message was sent to, which can be either the model element's unicast address or a subscription address of the model |
| 10-11 | uint16 | server_address | Address of the server model which sent the message |
| 12-15 | uint32 | remaining | Time (in milliseconds) remaining before transition from current state to target state is complete. Set to zero if no transition is taking place or if transition time does not apply to the message. |
| 16-17 | uint16 | flags | Message flags. It is a bitmask of the following values:<br>• **Bit 0:** Non-relayed. If non-zero indicates a response to a non-relayed request. |
| 18 | uint8 | type | Model-specific state type, identifying the kind of state reported in the status event. See get state types list for details. |
| 19 | uint8array | parameters | Message-specific parameters, serialized into a byte array |

**C Functions**

```
/* Event id */
gecko_evt_mesh_generic_client_server_status_id

/* Event structure */
struct gecko_msg_mesh_generic_client_server_status_evt_t
{
  uint16 model_id;,
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint32 remaining;,
  uint16 flags;,
  uint8 type;,
  uint8array parameters;
};
```

### 2.13.3 mesh_generic_client defines

#### 2.13.3.1 define_mesh_generic_client_get_state_type

Generic client get state type identifies the state which the client retrieves from the remote server model.

**Table 2.499. Defines**

| Value | Name | Description |
|---|---|---|
| 0 | MESH_GENERIC_CLIENT_state_on_off | Generic on/off get request |
| 1 | MESH_GENERIC_CLIENT_state_on_power_up | Generic on power up get request |
| 2 | MESH_GENERIC_CLIENT_state_level | Generic level get request |
| 3 | MESH_GENERIC_CLIENT_state_power_level | Generic power level get request |
| 4 | MESH_GENERIC_CLIENT_state_power_level_last | Generic power level last get request |
| 5 | MESH_GENERIC_CLIENT_state_power_level_default | Generic power level default get request |
| 6 | MESH_GENERIC_CLIENT_state_power_level_range | Generic power level range get request |
| 6 | MESH_GENERIC_CLIENT_state_transition_time | Generic transition time get request |
| 8 | MESH_GENERIC_CLIENT_state_battery | Generic battery get request |
| 9 | MESH_GENERIC_CLIENT_state_location_global | Generic global location get request |
| 10 | MESH_GENERIC_CLIENT_state_location_local | Generic local location get request |
| 11 | MESH_GENERIC_CLIENT_state_property_user | Generic user property get request |
| 12 | MESH_GENERIC_CLIENT_state_property_admin | Generic admin property get request |
| 13 | MESH_GENERIC_CLIENT_state_property_manuf | Generic manufacturer property get request |
| 14 | MESH_GENERIC_CLIENT_state_property_list_user | Generic user property list get request |
| 15 | MESH_GENERIC_CLIENT_state_property_list_admin | Generic admin property list get request |
| 16 | MESH_GENERIC_CLIENT_state_property_list_manuf | Generic manufacturer property list get request |
| 17 | MESH_GENERIC_CLIENT_state_property_list_client | Generic client property list get request |
| 128 | MESH_GENERIC_CLIENT_state_lightness_actual | Light actual lightness get request |
| 129 | MESH_GENERIC_CLIENT_state_lightness_linear | Light linear lightness get request |
| 130 | MESH_GENERIC_CLIENT_state_lightness_last | Light last lightness get request |
| 131 | MESH_GENERIC_CLIENT_state_lightness_default | Light default lightness get request |
| 132 | MESH_GENERIC_CLIENT_state_lightness_range | Light lightness range get request |
| 133 | MESH_GENERIC_CLIENT_state_ctl | Light lightness, color temperature, and delta UV server state identifier. Not to be used by client get requests. |
| 134 | MESH_GENERIC_CLIENT_state_ctl_temperature | Light color temperature and delta UV get request |
| 135 | MESH_GENERIC_CLIENT_state_ctl_default | Light lightness, color temperature, and delta UV default get request |
| 136 | MESH_GENERIC_CLIENT_state_ctl_range | Light color temperature range get request |
| 137 | MESH_GENERIC_CLIENT_state_ctl_lightness_temperature | Light lightness and color temperature get request. |
| 138 | MESH_GENERIC_CLIENT_state_hsl | Light lightness, color hue, and color saturation current value get request. |
| 139 | MESH_GENERIC_CLIENT_state_hsl_hue | Light color hue get request |
| 140 | MESH_GENERIC_CLIENT_state_hsl_saturation | Light color saturation get request |

| Value | Name | Description |
|---|---|---|
| 141 | MESH_GENERIC_CLIENT_state_hsl_default | Light lightness, color hue, and color saturation default get request |
| 142 | MESH_GENERIC_CLIENT_state_hsl_range | Light color hue and saturation range get request |
| 143 | MESH_GENERIC_CLIENT_state_hsl_target | Light lightness, color hue, and color saturation target value get request. |

#### 2.13.3.2 define_mesh_generic_client_set_request_type

Generic client set request type identifies the state which the client requests to be set to a new value on the remote server model.

**Table 2.500. Defines**

| Value | Name | Description |
|---|---|---|
| 0 | MESH_GENERIC_CLIENT_request_on_off | Generic on/off set request |
| 1 | MESH_GENERIC_CLIENT_request_on_power_up | Generic on power up set request |
| 2 | MESH_GENERIC_CLIENT_request_level | Generic level set request |
| 3 | MESH_GENERIC_CLIENT_request_level_delta | Generic level delta set request |
| 4 | MESH_GENERIC_CLIENT_request_level_move | Generic level move set request |
| 5 | MESH_GENERIC_CLIENT_request_level_halt | Generic level halt request |
| 6 | MESH_GENERIC_CLIENT_request_power_level | Generic power level set request |
| 7 | MESH_GENERIC_CLIENT_request_power_level_default | Generic power level default set request |
| 8 | MESH_GENERIC_CLIENT_request_power_level_range | Generic power level range set request |
| 9 | MESH_GENERIC_CLIENT_request_transition_time | Generic transition time set request |
| 10 | MESH_GENERIC_CLIENT_request_location_global | Generic global location set request |
| 11 | MESH_GENERIC_CLIENT_request_location_local | Generic local location set request |
| 12 | MESH_GENERIC_CLIENT_request_property_user | Generic user property set request |
| 13 | MESH_GENERIC_CLIENT_request_property_admin | Generic admin property set request |
| 14 | MESH_GENERIC_CLIENT_request_property_manuf | Generic manufacturer property set request |
| 128 | MESH_GENERIC_CLIENT_request_lightness_actual | Light actual lightness set request |
| 129 | MESH_GENERIC_CLIENT_request_lightness_linear | Light linear lightness set request |
| 130 | MESH_GENERIC_CLIENT_request_lightness_default | Light default lightness set request |
| 131 | MESH_GENERIC_CLIENT_request_lightness_range | Light lightness range set request |
| 132 | MESH_GENERIC_CLIENT_request_ctl | Light lightness, color temperature, and delta UV set request |
| 133 | MESH_GENERIC_CLIENT_request_ctl_temperature | Light color temperature and delta UV set request |
| 134 | MESH_GENERIC_CLIENT_request_ctl_default | Light lightness, color temperature, and delta UV default set request |
| 135 | MESH_GENERIC_CLIENT_request_ctl_range | Light color temperature range set request |
| 136 | MESH_GENERIC_CLIENT_request_hsl | Light lightness, color hue, and color saturation set request |
| 137 | MESH_GENERIC_CLIENT_request_hsl_hue | Light color hue set request |
| 138 | MESH_GENERIC_CLIENT_request_hsl_saturation | Light color saturation set request |
| 139 | MESH_GENERIC_CLIENT_request_hsl_default | Light lightness, color hue, and color saturation default set request |
| 140 | MESH_GENERIC_CLIENT_request_hsl_range | Light color hue and color saturation range set request |

## 2.14 Bluetooth Mesh Generic Server Model (mesh_generic_server)

Generic server model API provides functionality to send and receive messages using Bluetooth SIG server models, including generic server models and lighting server models.

Throughout the API the server model being used is identified by its element address and model ID, while the client model generating requests to the server model is identified by its element address and model ID.

The generic server model API is designed on the premise that the actual state the model represents resides in and is owned by the application, not by the stack.

The model acts as a cache for client queries, meaning that get state requests from client are handled automatically by the stack, and the application does not need to bother about those. The cached value is also used for periodic publication.

The flip side of caching is that when the state represented by the model changes in the application, it must update the cached value to the stack by issuing a server model update command.

When a client model requests a state change, the stack will generate a client request event which the application must process. Then, if the client needs a response the application has to issue a server response command corresponding to the request. Otherwise, the application only has to update the state with a server model update command, which does not result in sending any messages to the network.

Note that because the Mesh Model specification requires that certain states are bound together and because the stack enforces that, updating one cached state may result in an update of the corresponding bound state, for which the stack will generate a state changed event. An example of this is when a dimmable light is switched off and the lightness level, bound to the on/off state, is also set to zero because the states are bound.

Data for state change requests and server responses is passed as serialized byte arrays through BGAPI. There are functions to convert byte arrays to and from model state structures in the Bluetooth mesh SDK.

The stack will handle Mesh transaction layer segmentation and reassembly automatically if the messages sent are long enough to require it.

**Note on time resolution**

Because of message formatting, transition time and remaining time resolution units depend on the requested or reported value. For example, until 6.2 seconds it is 100 ms; until 62 seconds it is 1 s; until 620 seconds it is 10 s; and until 620 minutes it is 10 minutes. The value cannot be longer than 620 minutes. Therefore, it is not possible to request a delay of exactly 7500 ms. The resolution unit is 1 s between 6.2 and 62 seconds, so the value would be rounded down to 7 s.

Delay resolution is 5 ms and values will be rounded down to the closest 5 ms. The value can't be longer than 1275 ms.

### 2.14.1 mesh_generic_server commands

#### 2.14.1.1 cmd_mesh_generic_server_get_cached_state

Get model cached state. This command can be used to get cached model states after scene recall when using compacted recall events. This command supports only those states that would have been reported by @ref sl_btmesh_evt_generic_server_state_recall events.

**Table 2.501. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | uint16 | model_id | Server model ID |
| 8 | uint8 | type | Model-specific state type, identifying the kind of state reported in the state change event. See get state types list for details. |

**Table 2.502. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | remaining_ms | Time (in milliseconds) remaining before transition from current state to target state is complete. Ignored if no transition is taking place. |
| 10 | uint8array | parameters | Message-specific parameters, serialized into a byte array |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_generic_server_get_cached_state_rsp_t
*gecko_cmd_mesh_generic_server_get_cached_state(uint16 elem_index, uint16 model_id, uint8 type);

/* Response id */
gecko_rsp_mesh_generic_server_get_cached_state_id

/* Response structure */
struct gecko_msg_mesh_generic_server_get_cached_state_rsp_t
{
  uint16 result;,
  uint32 remaining_ms;,
  uint8array parameters;
};
```

#### 2.14.1.2 cmd_mesh_generic_server_init

Initialize generic server models. This command initializes all generic server models on the device. Alternatively, only the necessary server models can be initialized using model-specific initialization commands. Using model-specific initialization can result in a smaller firmware image size for SoC projects.

**Table 2.503. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x04 | method | Message ID |

**Table 2.504. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_rsp_t *gecko_cmd_mesh_generic_server_init();

/* Response id */
gecko_rsp_mesh_generic_server_init_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.3 cmd_mesh_generic_server_init_battery

Initialize generic battery server models

**Table 2.505. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0b | method | Message ID |

**Table 2.506. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_battery_rsp_t *gecko_cmd_mesh_generic_server_init_battery();

/* Response id */
gecko_rsp_mesh_generic_server_init_battery_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_battery_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.4 cmd_mesh_generic_server_init_common

Initialize generic server model common functionality. This should be called after all model-specific initialization calls are done, and does not need to be called if command to initialize all generic server models is used.

**Table 2.507. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x05 | method | Message ID |

**Table 2.508. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_common_rsp_t *gecko_cmd_mesh_generic_server_init_common();

/* Response id */
gecko_rsp_mesh_generic_server_init_common_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_common_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.5 cmd_mesh_generic_server_init_ctl

Initialize light CTL server models, light CTL temperature server models, light CTL setup server models, and all models they extend

**Table 2.509. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0f | method | Message ID |

**Table 2.510. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_ctl_rsp_t *gecko_cmd_mesh_generic_server_init_ctl();

/* Response id */
gecko_rsp_mesh_generic_server_init_ctl_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_ctl_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.6 cmd_mesh_generic_server_init_default_transition_time

Initialize generic default transition time server models

**Table 2.511. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x08 | method | Message ID |

**Table 2.512. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                    gecko_msg_mesh_generic_server_init_default_transition_time_rsp_t
*gecko_cmd_mesh_generic_server_init_default_transition_time();

/* Response id */
gecko_rsp_mesh_generic_server_init_default_transition_time_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_default_transition_time_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.7 cmd_mesh_generic_server_init_hsl

Initialize light HSL server models, light HSL hue server models, light HSL saturation server models, light HSL setup server models, and all models they extend

**Table 2.513. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x10 | method | Message ID |

**Table 2.514. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_hsl_rsp_t *gecko_cmd_mesh_generic_server_init_hsl();

/* Response id */
gecko_rsp_mesh_generic_server_init_hsl_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_hsl_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.8 cmd_mesh_generic_server_init_level

Initialize generic level server models

**Table 2.515. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x07 | method | Message ID |

**Table 2.516. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_level_rsp_t *gecko_cmd_mesh_generic_server_init_level();

/* Response id */
gecko_rsp_mesh_generic_server_init_level_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_level_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.9 cmd_mesh_generic_server_init_lightness

Initialize light lightness server models, light lightness setup server models, and all models they extend

**Table 2.517. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0e | method | Message ID |

**Table 2.518. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_lightness_rsp_t *gecko_cmd_mesh_generic_server_init_lightness();

/* Response id */
gecko_rsp_mesh_generic_server_init_lightness_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_lightness_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.10 cmd_mesh_generic_server_init_location

Initialize generic location and generic location setup server models

**Table 2.519. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0c | method | Message ID |

**Table 2.520. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_location_rsp_t *gecko_cmd_mesh_generic_server_init_location();

/* Response id */
gecko_rsp_mesh_generic_server_init_location_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_location_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.11 cmd_mesh_generic_server_init_on_off

Initialize generic on/off server models

**Table 2.521. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x06 | method | Message ID |

**Table 2.522. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_on_off_rsp_t *gecko_cmd_mesh_generic_server_init_on_off();

/* Response id */
gecko_rsp_mesh_generic_server_init_on_off_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_on_off_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.12 cmd_mesh_generic_server_init_power_level

Initialize generic power level server models, power level setup server models, and all models they extend

**Table 2.523. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0a | method | Message ID |

**Table 2.524. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_power_level_rsp_t *gecko_cmd_mesh_generic_server_init_power_level();

/* Response id */
gecko_rsp_mesh_generic_server_init_power_level_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_power_level_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.13 cmd_mesh_generic_server_init_power_on_off

Initialize generic power on/off server models, power on/off setup server models, and all models they extend

**Table 2.525. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x09 | method | Message ID |

**Table 2.526. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_generic_server_init_power_on_off_rsp_t
*gecko_cmd_mesh_generic_server_init_power_on_off();

/* Response id */
gecko_rsp_mesh_generic_server_init_power_on_off_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_power_on_off_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.14 cmd_mesh_generic_server_init_property

Initialize generic property server models

**Table 2.527. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0d | method | Message ID |

**Table 2.528. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_init_property_rsp_t *gecko_cmd_mesh_generic_server_init_property();

/* Response id */
gecko_rsp_mesh_generic_server_init_property_id

/* Response structure */
struct gecko_msg_mesh_generic_server_init_property_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.15 cmd_mesh_generic_server_publish

Publish server state into the network using the publish parameters configured into the model. The message is constructed using the cached state in the stack.

**Table 2.529. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | model_id | Server model ID |
| 6-7 | uint16 | elem_index | Server model element index |
| 8 | uint8 | type | Model-specific state type, identifying the kind of state used in the published message. See get state types list for details. |

**Table 2.530. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_generic_server_publish_rsp_t  *gecko_cmd_mesh_generic_server_publish(uint16  model_id,
uint16 elem_index, uint8 type);

/* Response id */
gecko_rsp_mesh_generic_server_publish_id

/* Response structure */
struct gecko_msg_mesh_generic_server_publish_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.16 cmd_mesh_generic_server_response

Server response to a client request. This command must be used when an application updates the server model state as a response to a client request event which required a response.

**Table 2.531. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | model_id | Server model ID |
| 6-7 | uint16 | elem_index | Server model element index |
| 8-9 | uint16 | client_address | Address of the client model which sent the message |
| 10-11 | uint16 | appkey_index | The application key index used |
| 12-15 | uint32 | remaining | Time (in milliseconds) remaining before transition from current state to target state is complete. Set to zero if no transition is taking place or if the transition time does not apply to the state change. |
| 16-17 | uint16 | flags | Message flags. Bitmask of the following:<br>• **Bit 0:** Non-relayed. If non-zero, indicates a response to a non-relayed request. |
| 18 | uint8 | type | Model-specific state type, identifying the kind of state to be updated. See get state types list for details. |
| 19 | uint8array | parameters | Message-specific parameters serialized into a byte array |

**Table 2.532. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_generic_server_response_rsp_t *gecko_cmd_mesh_generic_server_response(uint16 model_id,
uint16 elem_index, uint16 client_address, uint16 appkey_index, uint32 remaining, uint16 flags, uint8 type,
uint8 parameters_len, const uint8 *parameters_data);

/* Response id */
gecko_rsp_mesh_generic_server_response_id

/* Response structure */
```

```
struct gecko_msg_mesh_generic_server_response_rsp_t
{
  uint16 result;
};
```

#### 2.14.1.17 cmd_mesh_generic_server_update

Server state update. This command must be used when an application updates the server model state as a response to a client request event which did not require a response, but also when the application state changes spontaneously or as a result of some external (non-Mesh) event.

**Table 2.533. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | model_id | Server model ID |
| 6-7 | uint16 | elem_index | Server model element index |
| 8-11 | uint32 | remaining | Time (in milliseconds) remaining before transition from current state to target state is complete. Set to zero if no transition is taking place or if transition time does not apply to the state change. |
| 12 | uint8 | type | Model-specific state type, identifying the kind of state to be updated. See get state types list for details. |
| 13 | uint8array | parameters | Message-specific parameters, serialized into a byte array |

**Table 2.534. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_generic_server_update_rsp_t  *gecko_cmd_mesh_generic_server_update(uint16  model_id,
uint16 elem_index, uint32 remaining, uint8 type, uint8 parameters_len, const uint8 *parameters_data);

/* Response id */
gecko_rsp_mesh_generic_server_update_id

/* Response structure */
struct gecko_msg_mesh_generic_server_update_rsp_t
{
  uint16 result;
};
```

### 2.14.2 mesh_generic_server events

#### 2.14.2.1 evt_mesh_generic_server_client_request

State change request sent by a client model. This may be generated either because of a request directly to this model, or a request sent to an address which is subscribed to by the model.

**Table 2.535. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x14 | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | model_id | Server model ID |
| 6-7 | uint16 | elem_index | Server model element index |
| 8-9 | uint16 | client_address | Address of the client model which sent the message |
| 10-11 | uint16 | server_address | Address the message was sent to, which can be either the model element's unicast address, or model's subscription address |
| 12-13 | uint16 | appkey_index | The application key index used in encrypting the request; Responses need to be encrypted with the same key. |
| 14-17 | uint32 | transition | Requested transition time (in milliseconds) for the state change. If both the transition time and the delay are zero, the transition is immediate.<br>This applies to messages, which the Mesh Model specification defines to have transition and delay times and will be zero for others. |
| 18-19 | uint16 | delay | Delay time (in milliseconds) before starting the state change. If both the transition time and the delay are zero, the transition is immediate.<br>This applies to messages, which the Mesh Model specification defines to have transition and delay times and will be zero for others. |
| 20-21 | uint16 | flags | Message flags. Bitmask of the following values:<br>• **Bit 0:** Non-relayed. If non-zero, indicates that the client message was not relayed (TTL was zero) and that the server is within direct radio range of the client.<br>• **Bit 1:** Response required. If non-zero, the client expects a response from the server. |
| 22 | uint8 | type | Model-specific request type. See set request types list for details. |
| 23 | uint8array | parameters | Message-specific parameters serialized into a byte array |

**C Functions**

```
/* Event id */
gecko_evt_mesh_generic_server_client_request_id

/* Event structure */
struct gecko_msg_mesh_generic_server_client_request_evt_t
{
  uint16 model_id;,
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint32 transition;,
  uint16 delay;,
```

```
  uint16 flags;,
  uint8 type;,
  uint8array parameters;
};
```

#### 2.14.2.2 evt_mesh_generic_server_state_changed

Cached model state changed. This may happen either as a direct result of model state update by the application, in which case the event can be ignored, or because the update of one model state resulted in an update of a bound model state according to the Mesh model specification. In this case, the application should take action to update its own value accordingly.

**Table 2.536. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | model_id | Server model ID |
| 6-7 | uint16 | elem_index | Server model element index |
| 8-11 | uint32 | remaining | Time (in milliseconds) remaining before transition from current state to target state is complete. Ignored if no transition is taking place. |
| 12 | uint8 | type | Model-specific state type, identifying the kind of state reported in the state change event. See get state types list for details. |
| 13 | uint8array | parameters | Message-specific parameters, serialized into a byte array |

**C Functions**

```
/* Event id */
gecko_evt_mesh_generic_server_state_changed_id

/* Event structure */
struct gecko_msg_mesh_generic_server_state_changed_evt_t
{
  uint16 model_id;,
  uint16 elem_index;,
  uint32 remaining;,
  uint8 type;,
  uint8array parameters;
};
```

#### 2.14.2.3 evt_mesh_generic_server_state_recall

Cached model state changed due to scene recall operation.

**Table 2.537. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x1f | class | Message class: Bluetooth Mesh Generic Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | model_id | Server model ID |
| 6-7 | uint16 | elem_index | Server model element index |
| 8-11 | uint32 | transition_time | Time (in milliseconds) remaining before transition from current state to target state should be complete. Ignored if no transition is taking place. |
| 12 | uint8 | type | Model-specific state type, identifying the kind of state reported in the state change event. See get state types list for details. |
| 13 | uint8array | parameters | Model state - specific parameters, serialized into a byte array |

**C Functions**

```
/* Event id */
gecko_evt_mesh_generic_server_state_recall_id

/* Event structure */
struct gecko_msg_mesh_generic_server_state_recall_evt_t
{
  uint16 model_id;,
  uint16 elem_index;,
  uint32 transition_time;,
  uint8 type;,
  uint8array parameters;
};
```

## 2.15 Bluetooth Mesh Health Client Model (mesh_health_client)

Bluetooth mesh health client model functionality

### 2.15.1 mesh_health_client commands

#### 2.15.1.1 cmd_mesh_health_client_clear

Clear the fault status of a Health Server model or models in the network.

Besides the immediate result code, the response or responses (if the destination server address is a group address) from the network will generate server status report events.

**Table 2.538. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index. Identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address. May be a unicast address or a group address. |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |
| 10-11 | uint16 | vendor_id | Bluetooth vendor ID used in the request |
| 12 | uint8 | reliable | If non-zero, a reliable model message is used. |

**Table 2.539. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally (for instance, because of invalid parameters), an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_health_client_clear_rsp_t *gecko_cmd_mesh_health_client_clear(uint16 elem_index, uint16
server_address, uint16 appkey_index, uint16 vendor_id, uint8 reliable);

/* Response id */
gecko_rsp_mesh_health_client_clear_id

/* Response structure */
struct gecko_msg_mesh_health_client_clear_rsp_t
{
  uint16 result;
};
```

#### 2.15.1.2 cmd_mesh_health_client_get

Get the registered fault status of a Health Server model or models in the network.

Besides the immediate result code, the response or responses (if the destination server address is a group address) from the network will generate server status report events.

**Table 2.540. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index. Identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address. May be a unicast address or a group address. |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |
| 10-11 | uint16 | vendor_id | Bluetooth vendor ID used in the request |

**Table 2.541. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally, (for instance, because of invalid parameters) an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_health_client_get_rsp_t  *gecko_cmd_mesh_health_client_get(uint16  elem_index,  uint16
server_address, uint16 appkey_index, uint16 vendor_id);

/* Response id */
gecko_rsp_mesh_health_client_get_id

/* Response structure */
struct gecko_msg_mesh_health_client_get_rsp_t
{
  uint16 result;
};
```

#### 2.15.1.3 cmd_mesh_health_client_get_attention

Get the attention timer value of a Health Server model or models in the network.

Besides the immediate result code, the response or responses (if the destination server address is a group address) from the network will generate server status report events.

**Table 2.542. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index. Identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address. May be a unicast address or a group address. |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |

**Table 2.543. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally (for instance, because of invalid parameters), an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_health_client_get_attention_rsp_t  *gecko_cmd_mesh_health_client_get_attention(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_health_client_get_attention_id

/* Response structure */
struct gecko_msg_mesh_health_client_get_attention_rsp_t
{
  uint16 result;
};
```

#### 2.15.1.4 cmd_mesh_health_client_get_period

Get the health period log of a Health Server model or models in the network.

Except for the immediate result code, the response or responses (if the destination server address is a group address) from the network will generate server status report events.

**Table 2.544. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index. Identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address, which may be a unicast address or a group address |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |

**Table 2.545. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally, (for instance, because of invalid parameters) an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_health_client_get_period_rsp_t    *gecko_cmd_mesh_health_client_get_period(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_health_client_get_period_id

/* Response structure */
struct gecko_msg_mesh_health_client_get_period_rsp_t
{
  uint16 result;
};
```

#### 2.15.1.5 cmd_mesh_health_client_set_attention

Set the attention timer value of a Health Server model or models in the network.

Except for the immediate result code, the response or responses (if the destination server address is a group address) from the network will generate server status report events.

**Table 2.546. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index. Identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address. May be a unicast address or a group address. |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |
| 10 | uint8 | attention | Attention timer period in seconds |
| 11 | uint8 | reliable | If non-zero, a reliable model message is used. |

**Table 2.547. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally (for instance, because of invalid parameters), an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_health_client_set_attention_rsp_t  *gecko_cmd_mesh_health_client_set_attention(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 attention, uint8 reliable);

/* Response id */
gecko_rsp_mesh_health_client_set_attention_id

/* Response structure */
struct gecko_msg_mesh_health_client_set_attention_rsp_t
{
  uint16 result;
};
```

#### 2.15.1.6 cmd_mesh_health_client_set_period

Set the health period divisor of a Health Server model or models in the network.

Except for the immediate result code, the response or responses (if the destination server address is a group address) from the network will generate server status report events.

**Table 2.548. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index, which identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address. May be a unicast address or a group address. |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |
| 10 | uint8 | period | Health period divisor value |
| 11 | uint8 | reliable | If non-zero, a reliable model message is used. |

**Table 2.549. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally (for instance, because of invalid parameters), an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_health_client_set_period_rsp_t    *gecko_cmd_mesh_health_client_set_period(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 period, uint8 reliable);

/* Response id */
gecko_rsp_mesh_health_client_set_period_id

/* Response structure */
struct gecko_msg_mesh_health_client_set_period_rsp_t
{
  uint16 result;
};
```

#### 2.15.1.7 cmd_mesh_health_client_test

Execute a self test on a server model or models in the network.

**Table 2.550. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index. Identifies the client model used for sending the request. |
| 6-7 | uint16 | server_address | Destination server model address. May be a unicast address or a group address. |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request |
| 10 | uint8 | test_id | Test ID used in the request |
| 11-12 | uint16 | vendor_id | Bluetooth vendor ID used in the request |
| 13 | uint8 | reliable | If non-zero, a reliable model message is used. |

**Table 2.551. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally (for instance, because of invalid parameters) an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_health_client_test_rsp_t  *gecko_cmd_mesh_health_client_test(uint16  elem_index,  uint16
server_address, uint16 appkey_index, uint8 test_id, uint16 vendor_id, uint8 reliable);

/* Response id */
gecko_rsp_mesh_health_client_test_id

/* Response structure */
struct gecko_msg_mesh_health_client_test_rsp_t
{
  uint16 result;
};
```

### 2.15.2 mesh_health_client events

#### 2.15.2.1 evt_mesh_health_client_server_status

Receiving a Health Server fault status message generates this event.

The Client model may receive a status message because:

- * it made a get request to which a Server model responded, or
- * it made a clear request to which a Server model responded, or
- * it made a test request to which a Server model responded.

**Table 2.552. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Response status. If an error occurs (e.g., request timeout), the parameters other than element index, client address, and server address are to be ignored. |
| 6-7 | uint16 | elem_index | Client model element index. Identifies the client model which received the status message. |
| 8-9 | uint16 | client_address | Destination address the message was sent to |
| 10-11 | uint16 | server_address | Address of the Server model which sent the message |
| 12 | uint8 | current | Whether the event lists current fault array or registered fault array |
| 13 | uint8 | test_id | Test ID |
| 14-15 | uint16 | vendor_id | Bluetooth vendor ID used in the request |
| 16 | uint8array | faults | Fault array. See the Bluetooth Mesh Profile specification for a list of defined fault IDs. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_health_client_server_status_id

/* Event structure */
struct gecko_msg_mesh_health_client_server_status_evt_t
{
  uint16 result;,
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint8 current;,
  uint8 test_id;,
  uint16 vendor_id;,
  uint8array faults;
};
```

#### 2.15.2.2 evt_mesh_health_client_server_status_attention

Receiving a Health Server attention status message generates this event.

**Table 2.553. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Response status. If an error occurs (e.g., request timeout), ignore the parameters other than element index, client address, and server address. |
| 6-7 | uint16 | elem_index | Client model element index. Identifies the client model which received the status message. |
| 8-9 | uint16 | client_address | Destination address the message was sent to |
| 10-11 | uint16 | server_address | Address of the Server model which sent the message |
| 12 | uint8 | attention | Current attention timer value in seconds |

**C Functions**

```
/* Event id */
gecko_evt_mesh_health_client_server_status_attention_id

/* Event structure */
struct gecko_msg_mesh_health_client_server_status_attention_evt_t
{
  uint16 result;,
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint8 attention;
};
```

#### 2.15.2.3 evt_mesh_health_client_server_status_period

Receiving a Health Server period status message generates this event.

**Table 2.554. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x1a | class | Message class: Bluetooth Mesh Health Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Response status. If an error occurs (e.g., request timeout), ignore the parameters other than element index, client address, and server address. |
| 6-7 | uint16 | elem_index | Client model element index. Identifies the client model, which received the status message. |
| 8-9 | uint16 | client_address | Destination address the message was sent to |
| 10-11 | uint16 | server_address | Address of the Server model which sent the message |
| 12 | uint8 | period | Health period divisor value |

**C Functions**

```
/* Event id */
gecko_evt_mesh_health_client_server_status_period_id

/* Event structure */
struct gecko_msg_mesh_health_client_server_status_period_evt_t
{
  uint16 result;,
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint8 period;
};
```

## 2.16 Bluetooth Mesh Health Server Model (mesh_health_server)

Bluetooth mesh health server model functionality.

### 2.16.1 mesh_health_server commands

#### 2.16.1.1 cmd_mesh_health_server_clear_fault

Clear fault condition on an element.

**Table 2.555. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element on which the fault is no longer occurring. |
| 6 | uint8 | id | Fault ID. See the Mesh Profile specification for IDs defined by the Bluetooth SIG. |

**Table 2.556. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_health_server_clear_fault_rsp_t    *gecko_cmd_mesh_health_server_clear_fault(uint16
elem_index, uint8 id);

/* Response id */
gecko_rsp_mesh_health_server_clear_fault_id

/* Response structure */
struct gecko_msg_mesh_health_server_clear_fault_rsp_t
{
  uint16 result;
};
```

#### 2.16.1.2 cmd_mesh_health_server_set_fault

Set fault condition on an element.

**Table 2.557. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element on which the fault is occurring |
| 6 | uint8 | id | Fault ID. See the Mesh Profile specification for IDs defined by the Bluetooth SIG. |

**Table 2.558. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_health_server_set_fault_rsp_t *gecko_cmd_mesh_health_server_set_fault(uint16 elem_index,
uint8 id);

/* Response id */
gecko_rsp_mesh_health_server_set_fault_id

/* Response structure */
struct gecko_msg_mesh_health_server_set_fault_rsp_t
{
  uint16 result;
};
```

#### 2.16.1.3 cmd_mesh_health_server_test_response

Indicate to the stack that a test request has been completed and that the status may be communicated to the Health Client which made the test request.

**Table 2.559. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index. Identifies the Server model that received the request as well as the element on which the test is to be performed. |
| 6-7 | uint16 | client_address | Address of the client model which sent the message |
| 8-9 | uint16 | appkey_index | The application key index to use in encrypting the request. |
| 10-11 | uint16 | vendor_id | Bluetooth vendor ID used in the request |

**Table 2.560. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | If an error occurs locally (for instance, because of invalid parameters), an errorcode parameter is returned immediately. |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_health_server_test_response_rsp_t  *gecko_cmd_mesh_health_server_test_response(uint16
elem_index, uint16 client_address, uint16 appkey_index, uint16 vendor_id);

/* Response id */
gecko_rsp_mesh_health_server_test_response_id

/* Response structure */
struct gecko_msg_mesh_health_server_test_response_rsp_t
{
  uint16 result;
};
```

### 2.16.2 mesh_health_server events

#### 2.16.2.1 evt_mesh_health_server_attention

. Attention timer on an element is set to a given value. This may happen, for instance, during provisioning. The application should use suitable means to get the user's attention, e.g., by vibrating or blinking an LED.

**Table 2.561. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element for which attention timer was set |
| 6 | uint8 | timer | Timer value in seconds. If zero, user attention is no longer required. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_health_server_attention_id

/* Event structure */
struct gecko_msg_mesh_health_server_attention_evt_t
{
  uint16 elem_index;,
  uint8 timer;
};
```

#### 2.16.2.2 evt_mesh_health_server_test_request

. Health client request for a self test generates this event. After the test has been executed, test results may need to be reported.

**Table 2.562. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x1b | class | Message class: Bluetooth Mesh Health Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index. Identifies the Server model that received the request as well as the element on which the test is to be performed. |
| 6-7 | uint16 | client_address | Address of the client model which sent the message |
| 8-9 | uint16 | server_address | Destination address the message was sent to. It can be either the Server model element's unicast address, or a subscription address of the Server model. |
| 10-11 | uint16 | appkey_index | The application key index to use in encrypting the request. Any response sent must be encrypted using the same key. |
| 12 | uint8 | test_id | Test ID |
| 13-14 | uint16 | vendor_id | Bluetooth vendor ID used in the request |
| 15 | uint8 | response_required | Non-zero if client expects a response. The application should issue a Health Server test response command once it has processed the request. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_health_server_test_request_id

/* Event structure */
struct gecko_msg_mesh_health_server_test_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 test_id;,
  uint16 vendor_id;,
  uint8 response_required;
};
```

## 2.17 Bluetooth Mesh Light Control Client Model (mesh_lc_client)

Bluetooth Mesh LC Client model API provides functionality to send and receive messages to/from the LC Server and LC Setup Server models.

Throughout the API, the client model being used is identified by its element address and model ID, while the server model responding to client model requests is identified by its element address and model ID.

The API has functions for querying server model states and requesting server model state changes

### 2.17.1 mesh_lc_client commands

#### 2.17.1.1 cmd_mesh_lc_client_get_light_onoff

Get the Light OnOff status.

**Table 2.563. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |
| 6-7 | uint16 | server_address | Device to be queried. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | Appkey used by server_address. |

**Table 2.564. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_lc_client_get_light_onoff_rsp_t    *gecko_cmd_mesh_lc_client_get_light_onoff(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_lc_client_get_light_onoff_id

/* Response structure */
struct gecko_msg_mesh_lc_client_get_light_onoff_rsp_t
{
  uint16 result;
};
```

**Table 2.565. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_light_onoff_status | Event indicating an incoming LC Light OnOff Status message |

#### 2.17.1.2 cmd_mesh_lc_client_get_mode

Get the mode status.

**Table 2.566. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |
| 6-7 | uint16 | server_address | Device to be queried. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | Appkey used by server_address. |

**Table 2.567. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_client_get_mode_rsp_t  *gecko_cmd_mesh_lc_client_get_mode(uint16  elem_index,  uint16
server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_lc_client_get_mode_id

/* Response structure */
struct gecko_msg_mesh_lc_client_get_mode_rsp_t
{
  uint16 result;
};
```

**Table 2.568. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_mode_status | Event indicating an incoming LC Mode Status message |

#### 2.17.1.3 cmd_mesh_lc_client_get_om

Get the OM status.

**Table 2.569. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |
| 6-7 | uint16 | server_address | Device to be queried. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | Appkey used by server_address. |

**Table 2.570. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_client_get_om_rsp_t  *gecko_cmd_mesh_lc_client_get_om(uint16  elem_index,  uint16
server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_lc_client_get_om_id

/* Response structure */
struct gecko_msg_mesh_lc_client_get_om_rsp_t
{
  uint16 result;
};
```

**Table 2.571. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_om_status | Event indicating an incoming LC Occupancy Mode Status message |

#### 2.17.1.4 cmd_mesh_lc_client_get_property

Get the Property status.

**Table 2.572. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |
| 6-7 | uint16 | server_address | Device to be queried. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | Appkey used by server_address. |
| 10-11 | uint16 | property_id | The property ID to query. |

**Table 2.573. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_client_get_property_rsp_t  *gecko_cmd_mesh_lc_client_get_property(uint16  elem_index,
uint16 server_address, uint16 appkey_index, uint16 property_id);

/* Response id */
gecko_rsp_mesh_lc_client_get_property_id

/* Response structure */
struct gecko_msg_mesh_lc_client_get_property_rsp_t
{
  uint16 result;
};
```

**Table 2.574. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_property_status | Event indicating an incoming LC Property Status message |

#### 2.17.1.5 cmd_mesh_lc_client_init

Initializes the LC Client model. LC Client does not have any internal configuration, it only activates the model in the mesh stack.

**Table 2.575. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |

**Table 2.576. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_client_init_rsp_t *gecko_cmd_mesh_lc_client_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_lc_client_init_id

/* Response structure */
struct gecko_msg_mesh_lc_client_init_rsp_t
{
  uint16 result;
};
```

#### 2.17.1.6 cmd_mesh_lc_client_set_light_onoff

Set Light OnOff

**Table 2.577. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11 | uint8 | target_state | The target value of the Light LC Light OnOff state |
| 12 | uint8 | tid | Transaction identifier |
| 13-16 | uint32 | transition_time | Transition time in milliseconds. Value of 0xFFFFFFFF will cause this parameter as well as the "delay" parameter to be omitted. |
| 17-18 | uint16 | message_delay | Message execution delay in milliseconds. If the "transition_time" is 0xFFFFFFFF, this parameter is ignored. If both the transition time and the delay are zero the transition is immediate. |

**Table 2.578. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_lc_client_set_light_onoff_rsp_t    *gecko_cmd_mesh_lc_client_set_light_onoff(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint8 target_state, uint8 tid, uint32
transition_time, uint16 message_delay);

/* Response id */
gecko_rsp_mesh_lc_client_set_light_onoff_id

/* Response structure */
struct gecko_msg_mesh_lc_client_set_light_onoff_rsp_t
{
```

```
  uint16 result;
};
```

**Table 2.579. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_light_onoff_status | Event indicating an incoming LC Light OnOff Status message |

#### 2.17.1.7 cmd_mesh_lc_client_set_mode

Set mode

**Table 2.580. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11 | uint8 | mode | Mode value to set |

**Table 2.581. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_client_set_mode_rsp_t  *gecko_cmd_mesh_lc_client_set_mode(uint16  elem_index,  uint16
server_address, uint16 appkey_index, uint8 flags, uint8 mode);

/* Response id */
gecko_rsp_mesh_lc_client_set_mode_id

/* Response structure */
struct gecko_msg_mesh_lc_client_set_mode_rsp_t
{
  uint16 result;
};
```

**Table 2.582. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_mode_status | Event indicating an incoming LC Mode Status message |

#### 2.17.1.8 cmd_mesh_lc_client_set_om

Set occupancy mode

**Table 2.583. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11 | uint8 | mode | Mode value to set |

**Table 2.584. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_client_set_om_rsp_t  *gecko_cmd_mesh_lc_client_set_om(uint16  elem_index,  uint16
server_address, uint16 appkey_index, uint8 flags, uint8 mode);

/* Response id */
gecko_rsp_mesh_lc_client_set_om_id

/* Response structure */
struct gecko_msg_mesh_lc_client_set_om_rsp_t
{
  uint16 result;
};
```

**Table 2.585. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_om_status | Event indicating an incoming LC Occupancy Mode Status message |

#### 2.17.1.9 cmd_mesh_lc_client_set_property

Set a particular property

**Table 2.586. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11-12 | uint16 | property_id | Property ID for the LC Server. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | params | Byte array containing serialized fields of LC Property, excluding the property ID |

**Table 2.587. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_client_set_property_rsp_t *gecko_cmd_mesh_lc_client_set_property(uint16 elem_index,
uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id, uint8 params_len, const uint8
*params_data);

/* Response id */
gecko_rsp_mesh_lc_client_set_property_id

/* Response structure */
struct gecko_msg_mesh_lc_client_set_property_rsp_t
{
  uint16 result;
};
```

**Table 2.588. Events Generated**

| Event | Description |
|---|---|
| mesh_lc_client_property_status | Event indicating an incoming LC Property Status message |

### 2.17.2 mesh_lc_client events

#### 2.17.2.1 evt_mesh_lc_client_light_onoff_status

Event indicating an incoming LC Light OnOff Status message

**Table 2.589. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element for which received the status. |
| 6-7 | uint16 | server_address | Device which sent the status. |
| 8-9 | uint16 | destination_address | Address of the client or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by server_address. |
| 12 | uint8 | present_light_onoff | Present value of the Light OnOff state |
| 13 | uint8 | target_light_onoff | Target value of the Light OnOff state |
| 14-17 | uint32 | remaining_time | Time (in milliseconds) remaining in transition |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_client_light_onoff_status_id

/* Event structure */
struct gecko_msg_mesh_lc_client_light_onoff_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 present_light_onoff;,
  uint8 target_light_onoff;,
  uint32 remaining_time;
};
```

#### 2.17.2.2 evt_mesh_lc_client_mode_status

Event indicating an incoming LC Mode Status message

**Table 2.590. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element for which received the status. |
| 6-7 | uint16 | server_address | Device which sent the status. |
| 8-9 | uint16 | destination_address | Address of the client or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by server_address. |
| 12 | uint8 | mode_status_value | Value reported by server. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_client_mode_status_id

/* Event structure */
struct gecko_msg_mesh_lc_client_mode_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 mode_status_value;
};
```

#### 2.17.2.3 evt_mesh_lc_client_om_status

Event indicating an incoming LC Occupancy Mode Status message

**Table 2.591. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element for which received the status. |
| 6-7 | uint16 | server_address | Device which sent the status. |
| 8-9 | uint16 | destination_address | Address of the client or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by server_address. |
| 12 | uint8 | om_status_value | Value reported by server. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_client_om_status_id

/* Event structure */
struct gecko_msg_mesh_lc_client_om_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 om_status_value;
};
```

#### 2.17.2.4 evt_mesh_lc_client_property_status

Event indicating an incoming LC Property Status message

**Table 2.592. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x4c | class | Message class: Bluetooth Mesh Light Control Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element for which received the status. |
| 6-7 | uint16 | server_address | Device which sent the status. |
| 8-9 | uint16 | destination_address | Address of the client or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by server_address. |
| 12-13 | uint16 | property_id | Property ID |
| 14 | uint8array | property_value | Property value |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_client_property_status_id

/* Event structure */
struct gecko_msg_mesh_lc_client_property_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint16 property_id;,
  uint8array property_value;
};
```

## 2.18 Bluetooth Mesh Light Control Server Model (mesh_lc_server)

Bluetooth Mesh Light Control Server model functionality.

All LC Server state resides in and is own by the Model (stack). The state update notification events to the application are informational: the application is not required to react to them. The application may choose to save the LC Server state in persistent storage and set the states in the LC Server following a restart. To do this the application can utilize the notification events and update command.

Each LC Server instance requires that a Lightness Server is initialized in the element preceding the LC Server element: LC Server controls the Lightness Server residing in the preceding element. Each LC Server instance requires that a generic OnOff Server is initialized in the same element as the LC Server.

### 2.18.1 mesh_lc_server commands

#### 2.18.1.1 cmd_mesh_lc_server_deinit

De-initializes the LC Server model.

**Table 2.593. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.594. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_server_deinit_rsp_t *gecko_cmd_mesh_lc_server_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_lc_server_deinit_id

/* Response structure */
struct gecko_msg_mesh_lc_server_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.2 cmd_mesh_lc_server_get_lc_state

This command will fetch the current LC state. States can be as Off, Standby, Fade On, Run, Fade, Prolong, Fade Standby Auto, Fade Standby Manual

**Table 2.595. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.596. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | state | The current state of LC state machine" |
| 7-10 | uint32 | transition_time | Transtion time left for the current LC state. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_server_get_lc_state_rsp_t *gecko_cmd_mesh_lc_server_get_lc_state(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_lc_server_get_lc_state_id

/* Response structure */
struct gecko_msg_mesh_lc_server_get_lc_state_rsp_t
{
  uint16 result;,
  uint8 state;,
  uint32 transition_time;
};
```

#### 2.18.1.3 cmd_mesh_lc_server_init

Initializes the LC Server model. Server does not have any internal configuration, command only activates the model in the mesh stack.

Each LC Server instance requires that a Lightness Server is initialized in the element preceding the LC Server element: LC Server controls the Lightness Server residing in the preceding element. Each LC Server instance requires that a generic OnOff Server is initialized in the same element as the LC Server.

LC properties are initialized as follows:

PropertyID: PropertyValue 0x002B: 0x111111, 0x002C: 0x011111, 0x002D: 0x001111, 0x002E: 0xf000, 0x002F: 0x0f00, 0x0030: 0x00f0, 0x031: 50, 0x032: 25.0, 0x0033: 250.0, 0x0034: 80.0, 0x0035: 80.0, 0x0036: 3000, 0x0037: 3000, 0x0038: 3000, 0x0039: 3000, 0x003A: 0, 0x003B: 3000, 0x003C: 3000

PI Regulator interval (T) is initialized to 50ms

The rest of the state values are initialized to zero

**Table 2.597. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.598. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_server_init_rsp_t *gecko_cmd_mesh_lc_server_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_lc_server_init_id

/* Response structure */
struct gecko_msg_mesh_lc_server_init_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.4 cmd_mesh_lc_server_init_all_properties

Initialize all LC properties in one shot. Following values are used:

PropertyID: PropertyValue 0x002B: 0x111111, 0x002C: 0x011111, 0x002D: 0x001111, 0x002E: 0xf000, 0x002F: 0x0f00, 0x0030: 0x00f0, 0x031: 50, 0x032: 25.0, 0x0033: 250.0, 0x0034: 80.0, 0x0035: 80.0, 0x0036: 3000, 0x0037: 3000, 0x0038: 3000, 0x0039: 3000, 0x003A: 0, 0x003B: 3000, 0x003C: 3000

**Table 2.599. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.600. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_server_init_all_properties_rsp_t *gecko_cmd_mesh_lc_server_init_all_properties(uint16
elem_index);

/* Response id */
gecko_rsp_mesh_lc_server_init_all_properties_id

/* Response structure */
struct gecko_msg_mesh_lc_server_init_all_properties_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.5 cmd_mesh_lc_server_set_event_mask

This command will enable or disable additional diagnostics events. See lc_debug_events.

**Table 2.601. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |
| 6-7 | uint16 | event_type | The type of event to enable/disable. Options are:<br>lc_event_state_updated = 0x01, state_updated event report state changes lc_event_regulator_debug_info = 0x02, regulator_debug_info Regulator calculation details |
| 8 | uint8 | value | Valid values are 0 and 1 to disable or enable the event |

**Table 2.602. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_lc_server_set_event_mask_rsp_t    *gecko_cmd_mesh_lc_server_set_event_mask(uint16
elem_index, uint16 event_type, uint8 value);

/* Response id */
gecko_rsp_mesh_lc_server_set_event_mask_id

/* Response structure */
struct gecko_msg_mesh_lc_server_set_event_mask_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.6 cmd_mesh_lc_server_set_publish_mask

This command will update the bitmask that controls what messages are sent when the LC Server publishes. By default, the bitmask will be enabled to publish all three status messages.

**Table 2.603. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |
| 6-7 | uint16 | status_type | The type of status message to turn on/off. Options for this are:<br>LC Mode Status 0x8294 LC Occupancy Mode Status 0x8298 LC Light On Off Status 0x829C |
| 8 | uint8 | value | Turn on or off the status message. |

**Table 2.604. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_lc_server_set_publish_mask_rsp_t   *gecko_cmd_mesh_lc_server_set_publish_mask(uint16
elem_index, uint16 status_type, uint8 value);

/* Response id */
gecko_rsp_mesh_lc_server_set_publish_mask_id

/* Response structure */
struct gecko_msg_mesh_lc_server_set_publish_mask_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.7 cmd_mesh_lc_server_set_regulator_interval

This command will update the summation interval (T) at which the PI regulator is run. Only valid when regulator is disabled (Light LC Mode is 0).

**Table 2.605. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |
| 6 | uint8 | value | Valid values are 1ms-100ms. (Default: 50ms) |

**Table 2.606. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_lc_server_set_regulator_interval_rsp_t
*gecko_cmd_mesh_lc_server_set_regulator_interval(uint16 elem_index, uint8 value);

/* Response id */
gecko_rsp_mesh_lc_server_set_regulator_interval_id

/* Response structure */
struct gecko_msg_mesh_lc_server_set_regulator_interval_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.8 cmd_mesh_lc_server_update_light_onoff

Command for updating LC Server model Light OnOff state in the stack. Application may choose to directly set the model state in the stack, this function will pass the state value to the LC Server model.

**Table 2.607. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |
| 6 | uint8 | light_onoff | Light OnOff value |
| 7-10 | uint32 | transition_time_ms | Amount of time (in milliseconds) the element will take to transition to the target state from the present state. If set to 0 the transition will be immediate. |

**Table 2.608. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_server_update_light_onoff_rsp_t  *gecko_cmd_mesh_lc_server_update_light_onoff(uint16
elem_index, uint8 light_onoff, uint32 transition_time_ms);

/* Response id */
gecko_rsp_mesh_lc_server_update_light_onoff_id

/* Response structure */
struct gecko_msg_mesh_lc_server_update_light_onoff_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.9 cmd_mesh_lc_server_update_mode

Command for updating LC Server model Mode state in the stack. Application may choose to directly set the model state in the stack, this function will pass the state value to the LC Server model.

**Table 2.609. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |
| 6 | uint8 | mode | Mode value |

**Table 2.610. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_lc_server_update_mode_rsp_t  *gecko_cmd_mesh_lc_server_update_mode(uint16  elem_index,
uint8 mode);

/* Response id */
gecko_rsp_mesh_lc_server_update_mode_id

/* Response structure */
struct gecko_msg_mesh_lc_server_update_mode_rsp_t
{
  uint16 result;
};
```

#### 2.18.1.10 cmd_mesh_lc_server_update_om

Command for updating LC Server model Occupancy Mode state in the stack. Application may choose to directly set the model state in the stack, this function will pass the state value to the LC Server model.

**Table 2.611. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |
| 6 | uint8 | om | Occupancy Mode value |

**Table 2.612. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lc_server_update_om_rsp_t *gecko_cmd_mesh_lc_server_update_om(uint16 elem_index, uint8
om);

/* Response id */
gecko_rsp_mesh_lc_server_update_om_id

/* Response structure */
struct gecko_msg_mesh_lc_server_update_om_rsp_t
{
  uint16 result;
};
```

### 2.18.2 mesh_lc_server events

#### 2.18.2.1 evt_mesh_lc_server_ambient_lux_level_updated

LC Ambient LuxLevel state has been updated. The update is triggered by a reception of a sensor message.

**Table 2.613. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6-7 | uint16 | source_address | Message source address |
| 8-9 | uint16 | destination_address | Message destination address |
| 10-11 | uint16 | appkey_index | Message appkey index |
| 12-15 | uint32 | ambient_lux_level_value | The updated value of the LC Ambient LuxLevel state |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_ambient_lux_level_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_ambient_lux_level_updated_evt_t
{
  uint16 elem_index;,
  uint16 source_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint32 ambient_lux_level_value;
};
```

#### 2.18.2.2 evt_mesh_lc_server_light_onoff_updated

LC Light OnOff state has been updated. The update could be triggered by a reception of a client message or by an LC Server State Machine action.

**Table 2.614. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6-7 | uint16 | source_address | Message source address if triggered by a message, 0 otherwise. |
| 8-9 | uint16 | destination_address | Message destination address if triggered by a message, 0 otherwise. |
| 10-11 | uint16 | appkey_index | Message appkey index if triggered by a message, 0xFFFF otherwise. |
| 12 | uint8 | onoff_state | The target value of the Light LC Light OnOff state. |
| 13-16 | uint32 | onoff_trans_time | Amount of time (in milliseconds) the element will take to transition to the target state from the present state. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_light_onoff_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_light_onoff_updated_evt_t
{
  uint16 elem_index;,
  uint16 source_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 onoff_state;,
  uint32 onoff_trans_time;
};
```

#### 2.18.2.3 evt_mesh_lc_server_linear_output_updated

LC Linear Output state has been updated. The update is triggered by an LC Server State Machine action.

**Table 2.615. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6-7 | uint16 | linear_output_value | The updated value of the LC Linear Output state |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_linear_output_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_linear_output_updated_evt_t
{
  uint16 elem_index;,
  uint16 linear_output_value;
};
```

#### 2.18.2.4 evt_mesh_lc_server_mode_updated

LC Mode state has been updated. The update could be triggered by a reception of a client message or by an LC Server State Machine action.

**Table 2.616. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6-7 | uint16 | client_address | Message source address if triggered by a message, 0 otherwise. |
| 8-9 | uint16 | destination_address | Message destination address if triggered by a message, 0 otherwise. |
| 10-11 | uint16 | appkey_index | Message appkey index if triggered by a message, 0xFFFF otherwise. |
| 12 | uint8 | mode_value | The value the LC Mode state is being set to. |
| 13 | uint8 | manual_override | Light Control Mode state can be set to zero by a binding from Light Lightness Linear when it is modified by an action from the application or a Light Lightness Client command. In this case the parameter is set to 0x01. In all other cases, this parameter is zero. For example, when LC Mode is modified by the application or by a LC Client command, this parameter will be set to 0. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_mode_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_mode_updated_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 mode_value;,
  uint8 manual_override;
};
```

#### 2.18.2.5 evt_mesh_lc_server_occupancy_updated

LC Occupancy state has been updated. The update could be triggered by a reception of a sensor message or by an LC Server State Machine action.

**Table 2.617. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6-7 | uint16 | source_address | Message source address if triggered by a message, 0 otherwise. |
| 8-9 | uint16 | destination_address | Message destination address if triggered by a message, 0 otherwise. |
| 10-11 | uint16 | appkey_index | Message appkey index if triggered by a message, 0xFFFF otherwise. |
| 12 | uint8 | occupancy_value | The updated value of the LC Occupancy state |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_occupancy_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_occupancy_updated_evt_t
{
  uint16 elem_index;,
  uint16 source_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 occupancy_value;
};
```

#### 2.18.2.6 evt_mesh_lc_server_om_updated

LC Occupancy Mode state has been updated. The update could be triggered by a reception of a client message or by an LC Server State Machine action.

**Table 2.618. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6-7 | uint16 | client_address | Message source address if triggered by a message, 0 otherwise. |
| 8-9 | uint16 | destination_address | Message destination address if triggered by a message, 0 otherwise. |
| 10-11 | uint16 | appkey_index | Message appkey index if triggered by a message, 0xFFFF otherwise. |
| 12 | uint8 | om_value | The value the LC Occupancy Mode state is being set to. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_om_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_om_updated_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 om_value;
};
```

#### 2.18.2.7 evt_mesh_lc_server_regulator_debug_info

LC regulator calculation details

**Table 2.619. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where LC server is located |
| 6-7 | uint16 | i | Integral term |
| 8-9 | uint16 | l | Regulator output |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_regulator_debug_info_id

/* Event structure */
struct gecko_msg_mesh_lc_server_regulator_debug_info_evt_t
{
  uint16 elem_index;,
  uint16 i;,
  uint16 l;
};
```

#### 2.18.2.8 evt_mesh_lc_server_state_updated

LC state machine state has been updated. The update is triggered be LC mode being switched on or off and transitions between phases.

**Table 2.620. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x4d | class | Message class: Bluetooth Mesh Light Control Server Model |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element where the update happened |
| 6 | uint8 | state | The updated value of the LC state |
| 7-10 | uint32 | transition_time | Transition time defined for the current LC state. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_server_state_updated_id

/* Event structure */
struct gecko_msg_mesh_lc_server_state_updated_evt_t
{
  uint16 elem_index;,
  uint8 state;,
  uint32 transition_time;
};
```

### 2.18.3 mesh_lc_server enumerations

#### 2.18.3.1 enum_mesh_lc_server_lc_debug_events

These values identify optional diagnostic events that provide more information to the application about LC behavior

**Table 2.621. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | mesh_lc_server_lc_event_state_updated | Event reporting LC Server state machine state changes along with the remaining state timer |
| 2 | mesh_lc_server_lc_event_regulator_debug_info | Event reporting LC Server PI regulator integral term and regulator output |

#### 2.18.3.2 enum_mesh_lc_server_lc_state

These values define the possible states of Light Controller

**Table 2.622. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | mesh_lc_server_lc_state_off | The controller is turned off and does not control lighting. |
| 1 | mesh_lc_server_lc_state_standby | The controller is turned on and awaits an event from an occupancy sensor or a manual switch |
| 2 | mesh_lc_server_lc_state_fade_on | The controller has been triggered and gradually transitions to the Run phase, gradually dimming the lights up> |
| 3 | mesh_lc_server_lc_state_run | The lights are on and the timer counts down (but may be retriggered by a sensor or a switch event) |
| 4 | mesh_lc_server_lc_state_fade | The Run timer has expired and the controller gradually transitions to the Prolong state |
| 5 | mesh_lc_server_lc_state_prolong | The lights are at a lower level and the timer counts down (but may be retriggered by a sensor or a switch event) |
| 6 | mesh_lc_server_lc_state_fade_standby_auto | The controller gradually returns to the Standby state |
| 7 | mesh_lc_server_lc_state_fade_standby_manual | The controller gradually returns to the Standby state after external event |

## 2.19 Bluetooth Mesh Light Control Setup Server Model (mesh_lc_setup_server)

Bluetooth Mesh Light Control model Setup Server functionality.

This class provides the API that the LC Setup server uses to inform the application of its received events. The API is informational: application is not required to react to these events.

### 2.19.1 mesh_lc_setup_server commands

#### 2.19.1.1 cmd_mesh_lc_setup_server_update_property

Command for updating LC Server property. Application may choose to directly set model properties in the stack, this function will pass the property value to the LC Setup Server and on to the LC Server model.

**Table 2.623. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x4e | class | Message class: Bluetooth Mesh Light Control Setup Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | property_id | Property ID for the LC Server. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 8 | uint8array | params | Byte array containing serialized fields of LC Property, excluding the property ID |

**Table 2.624. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4e | class | Message class: Bluetooth Mesh Light Control Setup Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_lc_setup_server_update_property_rsp_t
*gecko_cmd_mesh_lc_setup_server_update_property(uint16 elem_index, uint16 property_id, uint8 params_len, const
uint8 *params_data);

/* Response id */
gecko_rsp_mesh_lc_setup_server_update_property_id

/* Response structure */
struct gecko_msg_mesh_lc_setup_server_update_property_rsp_t
{
  uint16 result;
};
```

### 2.19.2 mesh_lc_setup_server events

#### 2.19.2.1 evt_mesh_lc_setup_server_set_property

LC Property Set from the Client

**Table 2.625. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x4e | class | Message class: Bluetooth Mesh Light Control Setup Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the command. |
| 6-7 | uint16 | client_address | Device which sent the request. |
| 8-9 | uint16 | destination_address | Address of the server or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |
| 12-13 | uint16 | property_id | Property ID |
| 14 | uint8array | property_value | Property value |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lc_setup_server_set_property_id

/* Event structure */
struct gecko_msg_mesh_lc_setup_server_set_property_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint16 property_id;,
  uint8array property_value;
};
```

## 2.20 Bluetooth Mesh Low Power Node API (mesh_lpn)

These commands and events are for low-power operation, available in nodes which have the LPN feature.

### 2.20.1 mesh_lpn commands

#### 2.20.1.1 cmd_mesh_lpn_config

Configure the parameters for friendship establishment and LPN behavior.

**Table 2.626. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | setting_id | Identifies the LPN setting to be updated. |
| 5-8 | uint32 | value | New value for the given setting |

**Table 2.627. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lpn_config_rsp_t *gecko_cmd_mesh_lpn_config(uint8 setting_id, uint32 value);

/* Response id */
gecko_rsp_mesh_lpn_config_id

/* Response structure */
struct gecko_msg_mesh_lpn_config_rsp_t
{
  uint16 result;
};
```

#### 2.20.1.2 (deprecated) cmd_mesh_lpn_configure

**Deprecated** and replaced by mesh_lpn_config command. Configure the parameters for friendship establishment.

**Table 2.628. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | queue_length | Minimum queue length the friend must support. Choose an appropriate value based on the expected message frequency and LPN sleep period, because messages that do not fit into the friend queue are dropped. Note that the given value is rounded up to the nearest power of 2. Range: 2..128 |
| 5-8 | uint32 | poll_timeout | Poll timeout in milliseconds, which is the longest time that LPN sleeps in between querying its friend for queued messages. Long poll timeout allows the LPN to sleep for longer periods, at the expense of increased latency for receiving messages. Note that the given value is rounded up to the nearest 100 ms Range: 1 s to 95 h 59 min 59 s 900 ms |

**Table 2.629. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_lpn_configure_rsp_t   *gecko_cmd_mesh_lpn_configure(uint8   queue_length,   uint32
poll_timeout);

/* Response id */
gecko_rsp_mesh_lpn_configure_id

/* Response structure */
struct gecko_msg_mesh_lpn_configure_rsp_t
{
  uint16 result;
};
```

#### 2.20.1.3 cmd_mesh_lpn_deinit

Deinitialize the LPN functionality. After calling this command, a possible friendship with a Friend node is terminated and the node can operate in the network independently. After calling this command, do not call any other command in this class before the Low Power mode is initialized again.

**Table 2.630. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x01 | method | Message ID |

**Table 2.631. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lpn_deinit_rsp_t *gecko_cmd_mesh_lpn_deinit();

/* Response id */
gecko_rsp_mesh_lpn_deinit_id

/* Response structure */
struct gecko_msg_mesh_lpn_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.20.1.4 cmd_mesh_lpn_establish_friendship

Establish a friendship. After a friendship has been established the node can start saving power.

**Table 2.632. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | netkey_index | Network key index used in friendship request |

**Table 2.633. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_lpn_establish_friendship_rsp_t    *gecko_cmd_mesh_lpn_establish_friendship(uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_lpn_establish_friendship_id

/* Response structure */
struct gecko_msg_mesh_lpn_establish_friendship_rsp_t
{
  uint16 result;
};
```

#### 2.20.1.5 cmd_mesh_lpn_init

Initialize the Low Power node (LPN) mode. The node needs to be provisioned before calling this command. After the LPN mode is initialized, the node can't operate in the network without a Friend node. To establish a friendship with a nearby Friend node, use the establish friendship command. Make this call before calling the other commands in this class.

**Table 2.634. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x00 | method | Message ID |

**Table 2.635. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lpn_init_rsp_t *gecko_cmd_mesh_lpn_init();

/* Response id */
gecko_rsp_mesh_lpn_init_id

/* Response structure */
struct gecko_msg_mesh_lpn_init_rsp_t
{
  uint16 result;
};
```

#### 2.20.1.6 cmd_mesh_lpn_poll

Poll the Friend node for stored messages and security updates. This command may be used if the application is expecting to receive messages at a specific time. However, it is not required for correct operation, because the procedure will be performed autonomously before the poll timeout expires.

**Table 2.636. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x04 | method | Message ID |

**Table 2.637. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lpn_poll_rsp_t *gecko_cmd_mesh_lpn_poll();

/* Response id */
gecko_rsp_mesh_lpn_poll_id

/* Response structure */
struct gecko_msg_mesh_lpn_poll_rsp_t
{
  uint16 result;
};
```

#### 2.20.1.7 cmd_mesh_lpn_terminate_friendship

Terminate an already established friendship. Friendship terminated event will be emitted when the friendship termination has been completed.

**Table 2.638. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x05 | method | Message ID |

**Table 2.639. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_lpn_terminate_friendship_rsp_t *gecko_cmd_mesh_lpn_terminate_friendship();

/* Response id */
gecko_rsp_mesh_lpn_terminate_friendship_id

/* Response structure */
struct gecko_msg_mesh_lpn_terminate_friendship_rsp_t
{
  uint16 result;
};
```

### 2.20.2 mesh_lpn events

#### 2.20.2.1 evt_mesh_lpn_friendship_established

Indication that a friendship has been established

**Table 2.640. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | friend_address | Friend node address |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lpn_friendship_established_id

/* Event structure */
struct gecko_msg_mesh_lpn_friendship_established_evt_t
{
  uint16 friend_address;
};
```

#### 2.20.2.2 evt_mesh_lpn_friendship_failed

Indicates that friendship establishment failed.

**Table 2.641. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | reason | Reason for friendship establishment failure |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lpn_friendship_failed_id

/* Event structure */
struct gecko_msg_mesh_lpn_friendship_failed_evt_t
{
  uint16 reason;
};
```

#### 2.20.2.3 evt_mesh_lpn_friendship_terminated

Indicates that a friendship that was successfully established has been terminated.

**Table 2.642. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x23 | class | Message class: Bluetooth Mesh Low Power Node API |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | reason | Reason for friendship termination |

**C Functions**

```
/* Event id */
gecko_evt_mesh_lpn_friendship_terminated_id

/* Event structure */
struct gecko_msg_mesh_lpn_friendship_terminated_evt_t
{
  uint16 reason;
};
```

### 2.20.3 mesh_lpn enumerations

#### 2.20.3.1 enum_mesh_lpn_settings

Key values to identify LPN configurations

**Table 2.643. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | mesh_lpn_queue_length | Minimum queue length the friend must support. Choose an appropriate based on the expected message frequency and LPN sleep period, because messages that do not fit into the friend queue are dropped. Note that the given value is rounded up to the nearest power of 2. Range: 2..128 |
| 1 | mesh_lpn_poll_timeout | Poll timeout in milliseconds, which is the longest time that LPN sleeps in between querying its friend for queued messages. Long poll timeout allows the LPN to sleep for longer periods, at the expense of increased latency for receiving messages. Note that the given value is rounded up to the nearest 100 ms Range: 1 s to 95 h 59 min 59 s 900 ms |
| 2 | mesh_lpn_receive_delay | Receive delay in milliseconds. Receive delay is the time between the LPN sending a request and listening for a response. Receive delay allows the friend node time to prepare the message and LPN to sleep. Range: 10 ms to 255 ms The default receive delay in 10 ms. |
| 3 | mesh_lpn_request_retries | Request retry is the number of retry attempts to repeat e.g., the friend poll message if the friend update was not received by the LPN. Range is from 0 to 10, default is 3 |
| 4 | mesh_lpn_retry_interval | Time interval between retry attempts in milliseconds. Range is 0 to 100 ms. |
| 5 | mesh_lpn_clock_accuracy | Clock accuracy in ppm, which will be taken into account when opening and closing the receive window, and determining the poll timeout. Should be used with care, because inaccurate clock can increase the receive window lenght to up to 2,5 times in some cases. Default value is 0. |

## 2.21 Mesh Node (mesh_node)

Bluetooth mesh stack API for unprovisioned devices and provisioned nodes.

**Initialization:**

- Initialize node
-
- Node initialized

**Provisioning a node:**

- Get device UUID
- Start unprovisioned device beaconing
- Stop unprovisioned device beaconing
- Provisioning process has started
- Request to input out-of-band authentication data
- Respond to input out-of-band authentication request
- Request to display output out-of-band authentication data
- Request for static out-of-band authentication data
- Respond to static out-of-band authentication request
- Node has been provisioned
- Provisioning process has failed
- Pre-provision a device

**Node Configuration:**

- A cryptographic key has been added to the node
- Node-wide configuration has been queried
- Node-wide configuration has been modified
- Model configuration has been modified
- Received advertising events filter
- Factory reset mesh node

**Note on Bluetooth mesh addresses**

Bluetooth mesh address space is divided into sections containing ranges of addresses of various types. Different address types are used in different contexts. Some requests accept only certain address types.

The address types are as follows:

- **0x0000 Unassigned address:** represents an address that has not been set
- **0x0001..0x7fff Unicast addresses** are allocated by the Provisioner to provisioned nodes. Each element of a node has its own unicast address.
- **0x8000..0xbfff Virtual addresses** are 16-bit shorthand for 128-bit label UUIDs which are pre-allocated to specific purposes in relevant Bluetooth SIG specifications. Virtual addresses can typically be used in the same context as group addresses. Some commands require specifying the full label UUID instead of the virtual address shorthand.
- **0xc000..0xffef Group addresses** are allocated by the Provisioner for multicast communication.
- **0xfff0..0xffff Fixed group addresses** are allocated in the Mesh specification for multicast communication in a particular context. They can be used in the same context as regular group addresses. The following addresses are currently defined:
  - 0xfffc All-proxies broadcast address
  - 0xfffd All-friends broadcast address
  - 0xfffe All-relays broadcast address
  - 0xffff All-nodes broadcast address

### 2.21.1 mesh_node commands

#### 2.21.1.1 cmd_mesh_node_clear_statistics

**Table 2.644. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0a | method | Message ID |

**Table 2.645. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_clear_statistics_rsp_t *gecko_cmd_mesh_node_clear_statistics();

/* Response id */
gecko_rsp_mesh_node_clear_statistics_id

/* Response structure */
struct gecko_msg_mesh_node_clear_statistics_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.2 cmd_mesh_node_get_element_address

Get the unicast address configured to an element.

**Table 2.646. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |

**Table 2.647. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | address | The address of the element. Returns 0x0000 if the address is not configured or if an error occurs. |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_node_get_element_address_rsp_t    *gecko_cmd_mesh_node_get_element_address(uint16
elem_index);

/* Response id */
gecko_rsp_mesh_node_get_element_address_id

/* Response structure */
struct gecko_msg_mesh_node_get_element_address_rsp_t
{
  uint16 result;,
  uint16 address;
};
```

#### 2.21.1.3 cmd_mesh_node_get_ivrecovery_mode

Get current IV index recovery mode state. See set IV index recovery mode for details.

**Table 2.648. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x07 | method | Message ID |

**Table 2.649. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | mode | If non-zero, IV recovery is enabled. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_get_ivrecovery_mode_rsp_t *gecko_cmd_mesh_node_get_ivrecovery_mode();

/* Response id */
gecko_rsp_mesh_node_get_ivrecovery_mode_id

/* Response structure */
struct gecko_msg_mesh_node_get_ivrecovery_mode_rsp_t
{
  uint16 result;,
  uint8 mode;
};
```

#### 2.21.1.4 cmd_mesh_node_get_ivupdate_state

Get the current IV index update state in the network.

**Table 2.650. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0d | method | Message ID |

**Table 2.651. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | ivindex | Current IV index |
| 10 | uint8 | state | Indicates whether the IV index update is ongoing (1) or not (0). |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_get_ivupdate_state_rsp_t *gecko_cmd_mesh_node_get_ivupdate_state();

/* Response id */
gecko_rsp_mesh_node_get_ivupdate_state_id

/* Response structure */
struct gecko_msg_mesh_node_get_ivupdate_state_rsp_t
{
  uint16 result;,
  uint32 ivindex;,
  uint8 state;
};
```

#### 2.21.1.5 cmd_mesh_node_get_net_relay_delay

Get network relay delay interval. See set network relay delay command for details.

**Table 2.652. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0c | method | Message ID |

**Table 2.653. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | min | Minimum interval, in milliseconds |
| 7 | uint8 | max | Maximum interval, in milliseconds |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_get_net_relay_delay_rsp_t *gecko_cmd_mesh_node_get_net_relay_delay();

/* Response id */
gecko_rsp_mesh_node_get_net_relay_delay_id

/* Response structure */
struct gecko_msg_mesh_node_get_net_relay_delay_rsp_t
{
  uint16 result;,
  uint8 min;,
  uint8 max;
};
```

#### 2.21.1.6 cmd_mesh_node_get_seq_remaining

Get the number of sequence numbers remaining on an element (before sequence numbers are exhausted). Note that every node element keeps a separate sequence number counter.

**Table 2.654. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | elem_index | The index of queried element |

**Table 2.655. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | count | Remaining sequence number count |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_get_seq_remaining_rsp_t *gecko_cmd_mesh_node_get_seq_remaining(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_node_get_seq_remaining_id

/* Response structure */
struct gecko_msg_mesh_node_get_seq_remaining_rsp_t
{
  uint16 result;,
  uint32 count;
};
```

#### 2.21.1.7 cmd_mesh_node_get_statistics

**Table 2.656. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x09 | method | Message ID |

**Table 2.657. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | statistics | Raw statistics data |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_get_statistics_rsp_t *gecko_cmd_mesh_node_get_statistics();

/* Response id */
gecko_rsp_mesh_node_get_statistics_id

/* Response structure */
struct gecko_msg_mesh_node_get_statistics_rsp_t
{
  uint16 result;,
  uint8array statistics;
};
```

#### 2.21.1.8 cmd_mesh_node_get_uuid

Get the device UUID.

Every mesh device has a 128-bit UUID identifying the device. It is used primarily during provisioning, because it is broadcast in Unprovisioned Device Beacons to indicate that the device is ready to be provisioned.

This command can be used for debugging purposes. During provisioning the stack automatically uses the UUID of the device and it does not need to be explicitly specified when unprovisioned device beaconing is started.

**Table 2.658. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x03 | method | Message ID |

**Table 2.659. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | uuid | The 16-byte UUID of the device |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_get_uuid_rsp_t *gecko_cmd_mesh_node_get_uuid();

/* Response id */
gecko_rsp_mesh_node_get_uuid_id

/* Response structure */
struct gecko_msg_mesh_node_get_uuid_rsp_t
{
  uint16 result;,
  uint8array uuid;
};
```

#### 2.21.1.9 cmd_mesh_node_init

Initializes the Bluetooth mesh stack in Node role. When initialization is complete, a node initialized event will be generated.

This command must be issued before any other Bluetooth Mesh commands, except for command.

Note that you may initialize a device either in the Provisioner or the Node role, but not both.

**Table 2.660. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x00 | method | Message ID |

**Table 2.661. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_init_rsp_t *gecko_cmd_mesh_node_init();

/* Response id */
gecko_rsp_mesh_node_init_id

/* Response structure */
struct gecko_msg_mesh_node_init_rsp_t
{
  uint16 result;
};
```

**Table 2.662. Events Generated**

| Event | Description |
|---|---|
| mesh_node_initialized | Node is initialized and operational. |

#### 2.21.1.10 cmd_mesh_node_init_oob

Initialize the Bluetooth mesh stack in the Node role. When initialization is complete, a node initialized event is generated.

This command is the same as the node initialization command except for parameters defining whether OOB authentication data stored on the device can be used during provisioning.

This command must be issued before any other Bluetooth mesh commands, except for command.

Note that you may initialize a device either in the Provisioner or the Node role, but not both.

**Table 2.663. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | public_key | If non-zero, use the ECC key stored in the persistent store during provisioning instead of an ephemeral key. |
| 5 | uint8 | auth_methods | Allowed OOB authentication methods. The value is a bitmap so that multiple methods can be supported. |
| 6-7 | uint16 | output_actions | Allowed OOB Output Action types |
| 8 | uint8 | output_size | Maximum Output OOB size Valid values range from 0 (feature not supported) to 8. |
| 9-10 | uint16 | input_actions | Allowed OOB Input Action types |
| 11 | uint8 | input_size | Maximum Input OOB size. Valid values range from 0 (feature not supported) to 8. |
| 12-13 | uint16 | | Defines the OOB data location bitmask. |

**Table 2.664. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_init_oob_rsp_t *gecko_cmd_mesh_node_init_oob(uint8 public_key, uint8 auth_methods,
uint16 output_actions, uint8 output_size, uint16 input_actions, uint8 input_size, );

/* Response id */
gecko_rsp_mesh_node_init_oob_id
```

```
/* Response structure */
struct gecko_msg_mesh_node_init_oob_rsp_t
{
  uint16 result;
};
```

**Table 2.665. Events Generated**

| Event | Description |
|---|---|
| mesh_node_initialized | Node is initialized and operational. |

#### 2.21.1.11 cmd_mesh_node_input_oob_request_rsp

Provide the stack with the input out-of-band authentication data which the Provisioner is displaying.

**Table 2.666. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x02 | method | Message ID |
| 4 | uint8array | data | Raw 16-byte array containing the authentication data. |

**Table 2.667. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_node_input_oob_request_rsp_rsp_t   *gecko_cmd_mesh_node_input_oob_request_rsp(uint8
data_len, const uint8 *data_data);

/* Response id */
gecko_rsp_mesh_node_input_oob_request_rsp_id

/* Response structure */
struct gecko_msg_mesh_node_input_oob_request_rsp_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.12 cmd_mesh_node_request_ivupdate

Attempt to request an IV index update in the network.

Each network layer PDU that a node sends has a 24-bit sequence number attached to it. Each node element keeps a sequence number counter, which is incremented for every PDU sent out to the network. Repeating sequence numbers for a given IV index value is not allowed. As a result, if a node determines it is about to exhaust the available sequence numbers in one of its elements, it needs to request an IV index update by issuing this command.

Determining when a node may run out of sequence numbers has to be done at the application level because the stack can't determine how often the application plans to transmit to the network, i.e., how long the remaining sequence numbers might last.

See also the get remaining sequence numbers command.

Note that the call may fail for various reasons, for example if an IV index update is already ongoing, or if not enough time has passed since the previous IV index update.

**Table 2.668. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0e | method | Message ID |

**Table 2.669. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_request_ivupdate_rsp_t *gecko_cmd_mesh_node_request_ivupdate();

/* Response id */
gecko_rsp_mesh_node_request_ivupdate_id

/* Response structure */
struct gecko_msg_mesh_node_request_ivupdate_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.13 cmd_mesh_node_reset

Factory reset of the mesh node.

To complete procedure, the application should do its own cleanup duties and reset the hardware.

**Table 2.670. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x15 | method | Message ID |

**Table 2.671. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_reset_rsp_t *gecko_cmd_mesh_node_reset();

/* Response id */
gecko_rsp_mesh_node_reset_id

/* Response structure */
struct gecko_msg_mesh_node_reset_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.14 cmd_mesh_node_rssi

Get the latest RSSI value of a provisioned Bluetooth device.

The value indicates the best signal strength received from any node within the network. The value is cleared after calling this function meaning the next call will fail if no new RSSI value is received.

**Table 2.672. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x17 | method | Message ID |

**Table 2.673. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | int8 | rssi | Latest RSSI value. Units: dBm. Ignore this parameter if the command fails. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_rssi_rsp_t *gecko_cmd_mesh_node_rssi();

/* Response id */
gecko_rsp_mesh_node_rssi_id

/* Response structure */
struct gecko_msg_mesh_node_rssi_rsp_t
{
  uint16 result;,
  int8 rssi;
};
```

#### 2.21.1.15 cmd_mesh_node_save_replay_protection_list

Save the current replay protection list to the persistent store.

The replay protection list keeps track of the packet sequence numbers from different sources received by the node. The node will not process messages associated with already used sequence numbers and is therefore protected from replay attacks using previously recorded messages.

The replay protection list is kept in RAM during runtime. It needs to be saved to the persistent store periodically and always before the device powers off. Because the stack is not aware when this will happen, the application has to call this method while the node is getting ready to power down but is still running.

**Table 2.674. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x10 | method | Message ID |

**Table 2.675. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_node_save_replay_protection_list_rsp_t
*gecko_cmd_mesh_node_save_replay_protection_list();

/* Response id */
gecko_rsp_mesh_node_save_replay_protection_list_id

/* Response structure */
struct gecko_msg_mesh_node_save_replay_protection_list_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.16 cmd_mesh_node_set_adv_event_filter

Set filter for received advertising packet events.

As Mesh data traffic is carried over advertising events and the Bluetooth mesh stack is scanning continuously when the Bluetooth mesh stack is active, by default, the Bluetooth mesh stack filters out advertising events so that the application is not burdened by them.

If the application needs to process advertising events, it can use this command to unblock particular types of events.

**Table 2.676. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | mask | Enabled advertising packet type<br>• 0x01: Connectable undirected advertising<br>• 0x02: Scannable undirected advertising<br>• 0x04: Non connectable undirected advertising<br>• 0x08: Scan Response<br>• 0x8000: Use GAP data type. Don't use with other values |
| 6 | uint8array | gap_data_type | Used when the type is set to 0x8000. Events are generated when advertising packets contain any of the AD data types specified by this parameter. Type values are defined in the Bluetooth SIG Data Types Specification. Values must be set as a two digit hexadecimal number, maximum 8 items. |

**Table 2.677. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_node_set_adv_event_filter_rsp_t  *gecko_cmd_mesh_node_set_adv_event_filter(uint16  mask,
uint8 gap_data_type_len, const uint8 *gap_data_type_data);

/* Response id */
gecko_rsp_mesh_node_set_adv_event_filter_id

/* Response structure */
struct gecko_msg_mesh_node_set_adv_event_filter_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.17 cmd_mesh_node_set_beacon_reporting

Set secure network beaconing on or off. When on, every received secure network beacon will generate a beacon received event.

**Table 2.678. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x18 | method | Message ID |
| 4 | uint8 | report | Turn reporting on (nonzero) or off (zero). |

**Table 2.679. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_set_beacon_reporting_rsp_t *gecko_cmd_mesh_node_set_beacon_reporting(uint8 report);

/* Response id */
gecko_rsp_mesh_node_set_beacon_reporting_id

/* Response structure */
struct gecko_msg_mesh_node_set_beacon_reporting_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.18 cmd_mesh_node_set_iv_update_age

Set time since last IV update. After reboot the node does not know the time since the last IV update and assumes that it happend at the time of the reboot.

**Table 2.680. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x19 | method | Message ID |
| 4-7 | uint32 | age_s | Seconds since last IV update. Values from 0 to 345600 (96h) |

**Table 2.681. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_set_iv_update_age_rsp_t *gecko_cmd_mesh_node_set_iv_update_age(uint32 age_s);

/* Response id */
gecko_rsp_mesh_node_set_iv_update_age_id

/* Response structure */
struct gecko_msg_mesh_node_set_iv_update_age_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.19 cmd_mesh_node_set_ivrecovery_mode

Enable/disable IV index recovery mode.

If the node has not been in communication with the network for a long time (e.g., because it was turned off), it may have missed IV index updates and isn't able to communicate with other nodes. In this case, enable the IV index recovery mode.

**Table 2.682. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | mode | Zero to disable; non-zero to enable |

**Table 2.683. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_set_ivrecovery_mode_rsp_t *gecko_cmd_mesh_node_set_ivrecovery_mode(uint8 mode);

/* Response id */
gecko_rsp_mesh_node_set_ivrecovery_mode_id

/* Response structure */
struct gecko_msg_mesh_node_set_ivrecovery_mode_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.20 cmd_mesh_node_set_model_option

Set a model-specific option.

**Table 2.684. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |
| 10 | uint8 | option | Option to set. The following options are defined:<br>• **0x80** Generic level delta behavior. Used only with generic level models. |
| 11-14 | uint32 | value | Value for the option.<br>The following values are defined for generic level delta behavior option:<br>• **0x0** Generic level delta behavior: pass raw delta request data to application<br>• **0x1** Generic level delta behavior: pass processed delta request data to application (default) |

**Table 2.685. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_node_set_model_option_rsp_t  *gecko_cmd_mesh_node_set_model_option(uint16  elem_index,
uint16 vendor_id, uint16 model_id, uint8 option, uint32 value);

/* Response id */
gecko_rsp_mesh_node_set_model_option_id

/* Response structure */
struct gecko_msg_mesh_node_set_model_option_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.21.1.21 cmd_mesh_node_set_net_relay_delay

Set network relay delay interval.

This parameter determines the time a relay waits until it relays a network PDU. The value used is a random number within the specified interval.

Note that this value affects the first instance of the relayed network PDU. If relay retransmissions are enabled, the interval between retransmissions is defined by the relay state, set by the Provisioner of the network or by set local relay state test command.

**Table 2.686. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0b | method | Message ID |
| 4 | uint8 | min | Minimum interval, in milliseconds |
| 5 | uint8 | max | Maximum interval, in milliseconds, which must be equal to or greater than the minimum. |

**Table 2.687. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_set_net_relay_delay_rsp_t *gecko_cmd_mesh_node_set_net_relay_delay(uint8 min, uint8
max);

/* Response id */
gecko_rsp_mesh_node_set_net_relay_delay_id

/* Response structure */
struct gecko_msg_mesh_node_set_net_relay_delay_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.22 cmd_mesh_node_set_provisioning_data

Provision devices completely out-of-band. Populate the Provisioner's device database with the corresponding values to make the device reachable and configurable in the Provisioner's network.

See also the Provisioner command for adding a device to Provisioner's device database.

**NOTE**: The device must be reset after this command has been issued.

**Table 2.688. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x29 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x04 | method | Message ID |
| 4-19 | aes_key_128 | device_key | Device Key for this Device, shared by the Provisioner |
| 20-35 | aes_key_128 | network_key | Network key that the Provisioner has selected for this device |
| 36-37 | uint16 | netkey_index | Index of the Network Key the Provisioner has selected for this device |
| 38-41 | uint32 | iv_index | Current IV Index used in the network |
| 42-43 | uint16 | address | Address the Provisioner has allocated for this device's primary element |
| 44 | uint8 | kr_in_progress | Set to 1 if key refresh is currently in progress, otherwise 0. |

**Table 2.689. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_set_provisioning_data_rsp_t *gecko_cmd_mesh_node_set_provisioning_data(aes_key_128 device_key, aes_key_128 network_key, uint16 netkey_index, uint32 iv_index, uint16 address, uint8 kr_in_progress);

/* Response id */
gecko_rsp_mesh_node_set_provisioning_data_id

/* Response structure */
struct gecko_msg_mesh_node_set_provisioning_data_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.23 cmd_mesh_node_set_uuid

Write device UUID into the persistent store. This command must be called before initializing the Bluetooth mesh stack (before mesh_node_init or , otherwise the change will not take effect before a reboot.

Ensure that the UUID conforms to the format defined in RFC 4122

Note that UUID must not be changed when the device is provisioned to a network.

Furthermore, ensure that the UUID remains constant if a device has received a firmware update, which requires reprovisioning of the device after the update has been applied (e.g., new elements are added by the update).

**Table 2.690. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x11 | method | Message ID |
| 4-19 | uuid_128 | uuid | UUID to set |

**Table 2.691. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_set_uuid_rsp_t *gecko_cmd_mesh_node_set_uuid(uuid_128 uuid);

/* Response id */
gecko_rsp_mesh_node_set_uuid_id

/* Response structure */
struct gecko_msg_mesh_node_set_uuid_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.24 cmd_mesh_node_start_unprov_beaconing

Start sending Unprovisioned Device Beacons.

This command makes an unprovisioned device available for provisioning. The device will start sending periodic unprovisioned device beacons containing device UUID. It will also start listening for incoming Provisioner connection attempts on the specified bearers (PB-ADV, PB-GATT, or both). For PB-GATT, the device will also begin advertising its provisioning GATT service.

At the beginning of a provisioning process, a provisioning started event will be generated. When the device receives provisioning data from the Provisioner, a node provisioned event will be generated. If provisioning fails with an error, a provisioning failed event will be generated.

After it is provisioned, addresses are allocated for the node elements and a network key is deployed to the node, making the node ready for further configuration by the Provisioner. Note that at this point the node is not yet fully ready to communicate with other nodes on the network.

**Table 2.692. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | bearer | Bit mask for which bearer to use. Values are as follows:<br>• **1 (bit 0):** PB-ADV<br>• **2 (bit 1):** PB-GATT<br>Other bits are reserved and must not be used. |

**Table 2.693. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_node_start_unprov_beaconing_rsp_t  *gecko_cmd_mesh_node_start_unprov_beaconing(uint8
bearer);

/* Response id */
gecko_rsp_mesh_node_start_unprov_beaconing_id

/* Response structure */
struct gecko_msg_mesh_node_start_unprov_beaconing_rsp_t
{
  uint16 result;
};
```

**Table 2.694. Events Generated**

| Event | Description |
|---|---|
| mesh_node_provisioning_started | Provisioner has started provisioning this node. |
| mesh_node_provisioned | The node has received provisioning data (address allocation and a network key) from the Provisioner. A key added event will follow for the network key.<br><br>The node is now ready for further configuration by the Provisioner but is not yet ready for communication with other nodes in the network (it does not have any application keys and its models have not been set up). |
| mesh_node_provisioning_failed | Provisioning the node has failed. |

#### 2.21.1.25 cmd_mesh_node_static_oob_request_rsp

Provide the stack with static out-of-band authentication data which the stack requested.

**Table 2.695. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x13 | method | Message ID |
| 4 | uint8array | data | Raw 16-byte array containing the authentication data |

**Table 2.696. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_node_static_oob_request_rsp_rsp_t   *gecko_cmd_mesh_node_static_oob_request_rsp(uint8
data_len, const uint8 *data_data);

/* Response id */
gecko_rsp_mesh_node_static_oob_request_rsp_id

/* Response structure */
struct gecko_msg_mesh_node_static_oob_request_rsp_rsp_t
{
  uint16 result;
};
```

#### 2.21.1.26 cmd_mesh_node_stop_unprov_beaconing

Stop sending Unprovisioned Device Beacons.

**Table 2.697. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x16 | method | Message ID |

**Table 2.698. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_node_stop_unprov_beaconing_rsp_t *gecko_cmd_mesh_node_stop_unprov_beaconing();

/* Response id */
gecko_rsp_mesh_node_stop_unprov_beaconing_id

/* Response structure */
struct gecko_msg_mesh_node_stop_unprov_beaconing_rsp_t
{
  uint16 result;
};
```

### 2.21.2 mesh_node events

#### 2.21.2.1 evt_mesh_node_beacon_received

This event indicates reception of secure network beacon.

**Table 2.699. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key used to encrypt the beacon |
| 6 | uint8 | key_refresh | Indicates whether there is an ongoing key refresh. |
| 7 | uint8 | iv_update | Indicates whether there is an ongoing IV update. |
| 8-11 | uint32 | ivindex | IV index contained in the beacon. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_beacon_received_id

/* Event structure */
struct gecko_msg_mesh_node_beacon_received_evt_t
{
  uint16 netkey_index;,
  uint8 key_refresh;,
  uint8 iv_update;,
  uint32 ivindex;
};
```

#### 2.21.2.2 evt_mesh_node_changed_ivupdate_state

Network IV index update state has changed.

**Table 2.700. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0c | method | Message ID |
| 4-7 | uint32 | ivindex | Current IV index |
| 8 | uint8 | state | Indicates whether the IV index update is ongoing (1) or not (0). |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_changed_ivupdate_state_id

/* Event structure */
struct gecko_msg_mesh_node_changed_ivupdate_state_evt_t
{
  uint32 ivindex;,
  uint8 state;
};
```

#### 2.21.2.3 evt_mesh_node_config_get

Informative; Configuration Client requested the current value of a State in the Configuration Server Model.

**Table 2.701. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | id | Specifies to which State the command applies |
| 6-7 | uint16 | netkey_index | The network key index of the network to which the command applies. 0xffff for node-wide states. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_config_get_id

/* Event structure */
struct gecko_msg_mesh_node_config_get_evt_t
{
  uint16 id;,
  uint16 netkey_index;
};
```

#### 2.21.2.4 evt_mesh_node_config_set

Informative; Configuration Client changes the State in the Configuration Server Model.

**Table 2.702. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | id | Specifies to which state the command applies |
| 6-7 | uint16 | netkey_index | The network key index of the network to which the command applies. 0xffff for node-wide states. |
| 8 | uint8array | value | The new value |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_config_set_id

/* Event structure */
struct gecko_msg_mesh_node_config_set_evt_t
{
  uint16 id;,
  uint16 netkey_index;,
  uint8array value;
};
```

#### 2.21.2.5 evt_mesh_node_display_output_oob

Display output OOB data so Provisioner can input it.

**Table 2.703. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | output_action | Selected output action |
| 5 | uint8 | output_size | Size of data to output in characters. |
| 6 | uint8array | data | Raw 16-byte array containing the output data value. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_display_output_oob_id

/* Event structure */
struct gecko_msg_mesh_node_display_output_oob_evt_t
{
  uint8 output_action;,
  uint8 output_size;,
  uint8array data;
};
```

#### 2.21.2.6 evt_mesh_node_heartbeat

This event indicates reception of heartbeat message.

**Table 2.704. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | src_addr | Source address for the heartbeat message |
| 6-7 | uint16 | dst_addr | Destination address for the heartbeat message |
| 8 | uint8 | hops | Hops travelled by the heartbeat message |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_heartbeat_id

/* Event structure */
struct gecko_msg_mesh_node_heartbeat_evt_t
{
  uint16 src_addr;,
  uint16 dst_addr;,
  uint8 hops;
};
```

#### 2.21.2.7 evt_mesh_node_heartbeat_start

This event indicates start of heartbeat reception.

**Table 2.705. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | src_addr | Source address for the heartbeat message |
| 6-7 | uint16 | dst_addr | Destination address for the heartbeat message |
| 8-11 | uint32 | period | Heartbeat subscription period in seconds. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_heartbeat_start_id

/* Event structure */
struct gecko_msg_mesh_node_heartbeat_start_evt_t
{
  uint16 src_addr;,
  uint16 dst_addr;,
  uint32 period;
};
```

#### 2.21.2.8 evt_mesh_node_heartbeat_stop

This event indicates end of heartbeat reception.

**Table 2.706. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | src_addr | Source address for the heartbeat message |
| 6-7 | uint16 | dst_addr | Destination address for the heartbeat message |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_heartbeat_stop_id

/* Event structure */
struct gecko_msg_mesh_node_heartbeat_stop_evt_t
{
  uint16 src_addr;,
  uint16 dst_addr;
};
```

#### 2.21.2.9 evt_mesh_node_initialized

Node is initialized and operational.

**Table 2.707. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | provisioned | 1 if node is provisioned into a network, 0 if unprovisioned. |
| 5-6 | uint16 | address | Unicast address of the primary element of the node. Ignored if unprovisioned. Secondary elements have been assigned sequential unicast addresses following the primary element address. |
| 7-10 | uint32 | ivi | IV index for the first network of the node, ignore if unprovisioned. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_initialized_id

/* Event structure */
struct gecko_msg_mesh_node_initialized_evt_t
{
  uint8 provisioned;,
  uint16 address;,
  uint32 ivi;
};
```

#### 2.21.2.10 evt_mesh_node_input_oob_request

The Provisioner is displaying an out of band authentication value. Application on the node should provide the value to the Bluetooth mesh stack using the respond to input OOB request command.

**Table 2.708. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | input_action | Selected input action |
| 5 | uint8 | input_size | Size of data in the input in characters. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_input_oob_request_id

/* Event structure */
struct gecko_msg_mesh_node_input_oob_request_evt_t
{
  uint8 input_action;,
  uint8 input_size;
};
```

#### 2.21.2.11 evt_mesh_node_ivrecovery_needed

Network IV index recovery needed.

This event is generated when the node detects the network IV index is too far in the future to be automatically updated. See the IV recovery mode set command.

**Table 2.709. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0b | method | Message ID |
| 4-7 | uint32 | node_ivindex | Current IV index of the node |
| 8-11 | uint32 | network_ivindex | Received network IV index |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_ivrecovery_needed_id

/* Event structure */
struct gecko_msg_mesh_node_ivrecovery_needed_evt_t
{
  uint32 node_ivindex;,
  uint32 network_ivindex;
};
```

#### 2.21.2.12 evt_mesh_node_key_added

Received when a Configuration Client has deployed a new network or application key to the node.

**Table 2.710. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | type | Type of the new key. Values are as follows:<br>• **0x00:** Network key<br>• **0x01:** Application key |
| 5-6 | uint16 | index | Key index of the new key |
| 7-8 | uint16 | netkey_index | Network key index to which the application key is bound, which is ignored for network keys |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_key_added_id

/* Event structure */
struct gecko_msg_mesh_node_key_added_evt_t
{
  uint8 type;,
  uint16 index;,
  uint16 netkey_index;
};
```

#### 2.21.2.13 evt_mesh_node_key_removed

Received when a Configuration Client removes a network or application key from the node.

**Table 2.711. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0e | method | Message ID |
| 4 | uint8 | type | Type of the removed key. Values are as follows:<br>• **0x00:** Network key<br>• **0x01:** Application key |
| 5-6 | uint16 | index | Key index of the removed key |
| 7-8 | uint16 | netkey_index | Network key index to which the application key is bound, which is ignored for network keys |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_key_removed_id

/* Event structure */
struct gecko_msg_mesh_node_key_removed_evt_t
{
  uint8 type;,
  uint16 index;,
  uint16 netkey_index;
};
```

#### 2.21.2.14 evt_mesh_node_key_updated

Received when a Configuration Client updates a network or application key of the node.

**Table 2.712. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0f | method | Message ID |
| 4 | uint8 | type | Type of the updated key. Values are as follows:<br>• **0x00:** Network key<br>• **0x01:** Application key |
| 5-6 | uint16 | index | Key index of the updated key |
| 7-8 | uint16 | netkey_index | Network key index to which the application key is bound. Ignore this value if the event is for network key updates. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_key_updated_id

/* Event structure */
struct gecko_msg_mesh_node_key_updated_evt_t
{
  uint8 type;,
  uint16 index;,
  uint16 netkey_index;
};
```

#### 2.21.2.15 evt_mesh_node_model_config_changed

Informative. This event notifies that a remote Configuration Client has changed the configuration of a local model.

**Table 2.713. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | mesh_node_config_state | The configuration state which has changed. Values are as follows:<br>• **0x00:** Model application key bindings<br>• **0x01:** Model publication parameters<br>• **0x02:** Model subscription list |
| 5-6 | uint16 | element_address | Address of the element which contains the model |
| 7-8 | uint16 | vendor_id | Vendor ID of the model; value 0xffff is used for Bluetooth SIG models. |
| 9-10 | uint16 | model_id | Model ID of the model |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_model_config_changed_id

/* Event structure */
struct gecko_msg_mesh_node_model_config_changed_evt_t
{
  uint8 mesh_node_config_state;,
  uint16 element_address;,
  uint16 vendor_id;,
  uint16 model_id;
};
```

#### 2.21.2.16 evt_mesh_node_provisioned

The node has received provisioning data (address allocation and a network key) from the Provisioner. A key added event will follow for the network key.

The node is now ready for further configuration by the Provisioner but is not yet ready for communication with other nodes in the network (it does not have any application keys and its models have not been set up).

**Table 2.714. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x01 | method | Message ID |
| 4-7 | uint32 | iv_index | Current IV index of the provisioned network |
| 8-9 | uint16 | address | The unicast address that the Provisioner allocated for the primary element of the node. Secondary elements have been assigned sequentially following unicast addresses. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_provisioned_id

/* Event structure */
struct gecko_msg_mesh_node_provisioned_evt_t
{
  uint32 iv_index;,
  uint16 address;
};
```

#### 2.21.2.17 evt_mesh_node_provisioning_failed

Provisioning the node has failed.

**Table 2.715. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_provisioning_failed_id

/* Event structure */
struct gecko_msg_mesh_node_provisioning_failed_evt_t
{
  uint16 result;
};
```

#### 2.21.2.18 evt_mesh_node_provisioning_started

Provisioner has started provisioning this node.

**Table 2.716. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_provisioning_started_id

/* Event structure */
struct gecko_msg_mesh_node_provisioning_started_evt_t
{
  uint16 result;
};
```

#### 2.21.2.19 evt_mesh_node_reset

Provisioner has instructed the node to reset.

This event is generated when the Provisioner has ordered the node to be reset. Stack data has already been reset. This event is generated to inform the application that it should do its own cleanup duties and reset the hardware.

**Table 2.717. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0a | method | Message ID |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_reset_id

/* Event structure */
struct gecko_msg_mesh_node_reset_evt_t
{
};
```

#### 2.21.2.20 evt_mesh_node_static_oob_request

Static out of band authentication data is needed in the provisioning. The application on the node should provide the value to the Bluetooth mesh stack using the respond to static OOB authentication data request command.

**Table 2.718. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: Mesh Node |
| 3 | 0x0d | method | Message ID |

**C Functions**

```
/* Event id */
gecko_evt_mesh_node_static_oob_request_id

/* Event structure */
struct gecko_msg_mesh_node_static_oob_request_evt_t
{
};
```

### 2.21.3 mesh_node enumerations

#### 2.21.3.1 enum_mesh_node_auth_method_flag

Flags for supported OOB authentication methods during provisioning, which use a bitmap so that multiple methods can be supported.

**Table 2.719. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | mesh_node_auth_method_flag_none | Authentication without OOB is supported |
| 2 | mesh_node_auth_method_flag_static | Static OOB data authentication is supported |
| 4 | mesh_node_auth_method_flag_input | Input OOB authentication is supported |
| 8 | mesh_node_auth_method_flag_output | Output OOB authentication is supported |

#### 2.21.3.2 enum_mesh_node_config_state

Specify the state to which a Configuration Client/Server command/event applies.

**Table 2.720. Enumerations**

| Value | Name | Description |
|---|---|---|
| 32776 | mesh_node_dcd | Device Composition Data |
| 32777 | mesh_node_beacon | Status of broadcasting Secure Network Beacons |
| 32780 | mesh_node_default_ttl | Default Time-To-Live for messages |
| 32783 | mesh_node_friendship | Friend status |
| 32786 | mesh_node_gatt_proxy | GATT proxy status |
| 32789 | mesh_node_key_refresh | Key refresh status |
| 32803 | mesh_node_relay | Relay status |
| 32834 | mesh_node_identity | Identity status |
| 32804 | mesh_node_nettx | Network transmit status |

#### 2.21.3.3 enum_mesh_node_oob_input_action

Indicates the input OOB action selected by the Provisioner during provisioning of the device.

**Table 2.721. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | mesh_node_oob_input_action_push | Push a button on the device. |
| 1 | mesh_node_oob_input_action_twist | Twist a dial on the device. |
| 2 | mesh_node_oob_input_action_numeric | Input a numeric authentication code. |
| 3 | mesh_node_oob_input_action_alpha | Input an alphanumeric authentication code. |

#### 2.21.3.4 enum_mesh_node_oob_input_action_flag

Flags for supported input OOB actions during provisioning, which use a bitmap so that multiple actions can be supported.

**Table 2.722. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | mesh_node_oob_input_action_flag_push | Push a button on the device. |
| 2 | mesh_node_oob_input_action_flag_twist | Twist a dial on the device. |
| 4 | mesh_node_oob_input_action_flag_numeric | Input a numeric authentication code. |
| 8 | mesh_node_oob_input_action_flag_alpha | Input an alphanumeric authentication code. |

#### 2.21.3.5 enum_mesh_node_oob_output_action

Indicates the output OOB action selected by the Provisioner during provisioning of the device.

**Table 2.723. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | mesh_node_oob_output_action_blink | Blink a light. |
| 1 | mesh_node_oob_output_action_beep | Emit a sound. |
| 2 | mesh_node_oob_output_action_vibrate | Vibrate the device. |
| 3 | mesh_node_oob_output_action_numeric | Output a numeric authentication code. |
| 4 | mesh_node_oob_output_action_alpha | Output an alphanumeric authentication code. |

#### 2.21.3.6 enum_mesh_node_oob_output_action_flag

Flags for supported output OOB actions during provisioning, which use a bitmap so that multiple actions can be supported.

**Table 2.724. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | mesh_node_oob_output_action_flag_blink | Blink a light. |
| 2 | mesh_node_oob_output_action_flag_beep | Emit a sound. |
| 4 | mesh_node_oob_output_action_flag_vibrate | Vibrate the device. |
| 8 | mesh_node_oob_output_action_flag_numeric | Output a numeric authentication code. |
| 16 | mesh_node_oob_output_action_flag_alpha | Output an alphanumeric authentication code. |

## 2.22 Bluetooth Mesh Stack Provisioner (mesh_prov)

Bluetooth mesh stack API for the embedded Provisioner

Commands in this class provision nodes in the mesh network and generate security keys for the network.

**Initialization:**

- Initialize provisioner
- Provisioner initialized

**Provisioning a node:**

- Scan for unprovisioned device beacons
- Stop scanning for unprovisioned device beacons
- Unprovisioned device beacon seen
- URI advertisement seen
- Provision a device over PB-ADV
- Provision a device over PB-GATT
- Request to display input out-of-band data to the user to input on the node
- Request for out-of-band public key of a node
- Provide stack with out-of-band public key of a node
- Request for out-of-band authentication data of a node
- Provide stack with out-of-band authentication data of a node
- Device Provisioned
- Provisioning a device failed

**Key Management**

- Create a new network key on the Provisioner
- Create a new application key on the Provisioner
- Start a key refresh procedure
- Suspend an ongoing key refresh procedure
- Resume a suspended key refresh procedure
- Get node key refresh blacklist status
- Set node key refresh blacklist status
- Get node key refresh phase
-

**Device Database**

- Add a node to the device database
- Remove a node from the device database
- Fetch node data from the device database
- Request a list of nodes in the device database
- Device database list result
- Update default network key index for a device database entry

These commands are available only if the Provisioner functionality is compiled in the device. Otherwise, a "feature not implemented" error code will be returned for all functions in this class.

### 2.22.1 mesh_prov commands

#### 2.22.1.1 cmd_mesh_prov_abort_provisioning

Abort provisioning.

**Table 2.725. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x41 | method | Message ID |
| 4-19 | uuid_128 | uuid | UUID of the device being provisioned |
| 20 | uint8 | reason | Reason for aborting. Values are as follows:<br>• **1:** Invalid PDU<br>• **2:** Invalid PDU format<br>• **3:** Unexpected PDU<br>• **4:** Confirmation failed<br>• **5:** Out of resources<br>• **6:** Decryption failed<br>• **7:** Unexpected error<br>• **8:** Unable to assign address |

**Table 2.726. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x41 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_abort_provisioning_rsp_t  *gecko_cmd_mesh_prov_abort_provisioning(uuid_128  uuid,
uint8 reason);

/* Response id */
gecko_rsp_mesh_prov_abort_provisioning_id

/* Response structure */
struct gecko_msg_mesh_prov_abort_provisioning_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.2 (deprecated) cmd_mesh_prov_appkey_add

**Deprecated** and replaced by mesh_config_client_add_appkey command.

Push an application key to a node. The key must exist on the Provisioner (see create application key command).

An application key is always bound to a network key. In other words, the application key is only valid in the context of a particular network key. The selected network key must exist on the Provisioner (see create network key command) and must have been deployed on the node prior to this command (either during provisioning or with an add network key command).

Node response is reported with an configuration status event.

**Table 2.727. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | netkey_index | The network key index to which the application key is bound |
| 8-9 | uint16 | appkey_index | The index of the application key to push to the node |

**Table 2.728. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_prov_appkey_add_rsp_t   *gecko_cmd_mesh_prov_appkey_add(uint16   address,   uint16
netkey_index, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_prov_appkey_add_id

/* Response structure */
struct gecko_msg_mesh_prov_appkey_add_rsp_t
{
  uint16 result;
};
```

**Table 2.729. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.3 (deprecated) cmd_mesh_prov_appkey_delete

**Deprecated** and replaced by mesh_config_client_remove_appkey command.

Delete an application key on a node.

Note that the deleted key will be removed from any model bindings on the node at the same time automatically. There is no need to explicitly delete them using model-application key unbind command.

Node response is reported with an configuration status event.

**Table 2.730. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | netkey_index | Index of the network key to which the application key is bound on the node |
| 8-9 | uint16 | appkey_index | Index of the application key to delete |

**Table 2.731. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_appkey_delete_rsp_t  *gecko_cmd_mesh_prov_appkey_delete(uint16  address,  uint16
netkey_index, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_prov_appkey_delete_id

/* Response structure */
struct gecko_msg_mesh_prov_appkey_delete_rsp_t
{
  uint16 result;
};
```

**Table 2.732. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.4 (deprecated) cmd_mesh_prov_appkey_get

**Deprecated** and replaced by mesh_config_client_list_appkeys command.

Get a list of application keys bound to a network key on a node.

Return a list of application key indices for the application keys bound to a particular network key on a node.

Node response is reported with a number of application key listevents, terminated by a event.

**Table 2.733. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2a | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | netkey_index | Index of the network key to which the application keys are bound on the node |

**Table 2.734. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_appkey_get_rsp_t  *gecko_cmd_mesh_prov_appkey_get(uint16  address,  uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_appkey_get_id

/* Response structure */
struct gecko_msg_mesh_prov_appkey_get_rsp_t
{
  uint16 result;
};
```

**Table 2.735. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_appkey_list | **Deprecated** and replaced by mesh_config_client_appkey_list event.<br>Application key list event. |
| mesh_prov_appkey_list_end | **Deprecated** and replaced by mesh_config_client_appkey_list_end event.<br>Application key list terminator event. |

#### 2.22.1.5 cmd_mesh_prov_create_appkey

Creates a new application key on the Provisioner.

An application key is always bound to a network key. In other words, the application key is only valid in the context of a particular network key. The selected network key must exist on the Provisioner (see create network key command).

The created application key can be deployed on a node using the add application key command.

**Table 2.736. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key to which the application key will be bound |
| 6 | uint8array | key | Key value to use; set to zero-length array to generate random key. |

**Table 2.737. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | appkey_index | Index of the new application key. Ignore it if the result was non-zero. |
| 8 | uint8array | key | New application key. Ignore it if the result was non-zero. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_create_appkey_rsp_t *gecko_cmd_mesh_prov_create_appkey(uint16 netkey_index, uint8
key_len, const uint8 *key_data);

/* Response id */
gecko_rsp_mesh_prov_create_appkey_id

/* Response structure */
struct gecko_msg_mesh_prov_create_appkey_rsp_t
{
  uint16 result;,
  uint16 appkey_index;,
  uint8array key;
};
```

#### 2.22.1.6 cmd_mesh_prov_create_network

Create a new network key on the Provisioner.

The created key can be deployed on a node using the add network key command.

**Table 2.738. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x03 | method | Message ID |
| 4 | uint8array | key | Key value to use. Set to zero-length array to generate a random key. |

**Table 2.739. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | network_id | Index of the newly created network key |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_create_network_rsp_t *gecko_cmd_mesh_prov_create_network(uint8 key_len, const uint8
*key_data);

/* Response id */
gecko_rsp_mesh_prov_create_network_id

/* Response structure */
struct gecko_msg_mesh_prov_create_network_rsp_t
{
  uint16 result;,
  uint8 network_id;
};
```

#### 2.22.1.7 cmd_mesh_prov_ddb_add

Add a new node entry to the Provisioner's device database. Note that the device key, primary element address, and network key need to be deployed to the node being added to ensure it's configurable. See set node provisioning data command.

**Table 2.740. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x25 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x18 | method | Message ID |
| 4-19 | uuid_128 | uuid | UUID of the node to add |
| 20-35 | aes_key_128 | device_key | Device key value for the node |
| 36-37 | uint16 | netkey_index | Index of the network key the node will be used for configuration |
| 38-39 | uint16 | address | Unicast address to allocate for the node's primary element |
| 40 | uint8 | elements | Number of elements the device has |

**Table 2.741. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_ddb_add_rsp_t *gecko_cmd_mesh_prov_ddb_add(uuid_128 uuid, aes_key_128 device_key,
uint16 netkey_index, uint16 address, uint8 elements);

/* Response id */
gecko_rsp_mesh_prov_ddb_add_id

/* Response structure */
struct gecko_msg_mesh_prov_ddb_add_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.8 cmd_mesh_prov_ddb_delete

Delete node information from the Provisioner database. This should be followed by a key refresh procedure updating the keys of the remaining nodes to make sure the deleted node is shut off from the network.

**Table 2.742. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x17 | method | Message ID |
| 4-19 | uuid_128 | uuid | UUID of the node to delete |

**Table 2.743. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_ddb_delete_rsp_t *gecko_cmd_mesh_prov_ddb_delete(uuid_128 uuid);

/* Response id */
gecko_rsp_mesh_prov_ddb_delete_id

/* Response structure */
struct gecko_msg_mesh_prov_ddb_delete_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.9 cmd_mesh_prov_ddb_get

Get a Provisioner device database entry with a matching UUID.

**Note:** this command handles sensitive data, and in Secure NCP applications requires encryption to be enabled. Otherwise the error bg_err_application_mismatched_or_insufficient_security will be returned.

**Table 2.744. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x16 | method | Message ID |
| 4 | uint8array | uuid | UUID of the Device to retrieve |

**Table 2.745. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x17 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-21 | aes_key_128 | device_key | Device Key |
| 22-23 | uint16 | netkey_index | Index of the network key with which the node was initially provisioned. Used for network-level encryption of Configuration Client messages. |
| 24-25 | uint16 | address | Unicast address of the primary element of the node |
| 26 | uint8 | elements | Number of elements in the node |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_ddb_get_rsp_t *gecko_cmd_mesh_prov_ddb_get(uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_ddb_get_id

/* Response structure */
struct gecko_msg_mesh_prov_ddb_get_rsp_t
{
  uint16 result;,
  aes_key_128 device_key;,
  uint16 netkey_index;,
  uint16 address;,
  uint8 elements;
};
```

#### 2.22.1.10 cmd_mesh_prov_ddb_list_devices

Lists nodes known by this Provisioner. A number of database listing events will be generated.

**Table 2.746. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x19 | method | Message ID |

**Table 2.747. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | count | Number of events that will follow |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_ddb_list_devices_rsp_t *gecko_cmd_mesh_prov_ddb_list_devices();

/* Response id */
gecko_rsp_mesh_prov_ddb_list_devices_id

/* Response structure */
struct gecko_msg_mesh_prov_ddb_list_devices_rsp_t
{
  uint16 result;,
  uint16 count;
};
```

**Table 2.748. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_ddb_list | Provisioner's device database list entry |

#### 2.22.1.11 cmd_mesh_prov_ddb_update_netkey_index

Update a node's entry in the Provisioner's device database by setting a new value to the netkey_index field. The netkey_index field is used to determine the network key to use when encrypting and decrypting configuration model messages to and from the node.

**Table 2.749. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3b | method | Message ID |
| 4-19 | uuid_128 | uuid | UUID of the node |
| 20-21 | uint16 | netkey_index | Index of the network key used in configuring the node. |

**Table 2.750. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_ddb_update_netkey_index_rsp_t *gecko_cmd_mesh_prov_ddb_update_netkey_index(uuid_128
uuid, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_prov_ddb_update_netkey_index_id

/* Response structure */
struct gecko_msg_mesh_prov_ddb_update_netkey_index_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.12 cmd_mesh_prov_flush_key_refresh_state

Clear key refresh state stored in persistent storage.

Note that this command should not normally be used. It is intended only for clearing stored key refresh state when a key refresh procedure has been suspended and will not be resumed, either because the network key has been deleted from all nodes or the responsibility for completing the key refresh has been moved to another Provisioner.

**Table 2.751. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x40 | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key identifying a key refresh procedure |

**Table 2.752. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x40 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_flush_key_refresh_state_rsp_t  *gecko_cmd_mesh_prov_flush_key_refresh_state(uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_flush_key_refresh_state_id

/* Response structure */
struct gecko_msg_mesh_prov_flush_key_refresh_state_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.13 (deprecated) cmd_mesh_prov_friend_timeout_get

**Deprecated** and replaced by mesh_config_client_get_lpn_polltimeout command.

LPN poll timeout request. Result is reported in a friend poll timeout status event.

**Table 2.753. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x32 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the friend node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |
| 8-9 | uint16 | lpn_address | Unicast address of the LPN node |

**Table 2.754. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x32 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_friend_timeout_get_rsp_t  *gecko_cmd_mesh_prov_friend_timeout_get(uint16  address,
uint16 netkey_index, uint16 lpn_address);

/* Response id */
gecko_rsp_mesh_prov_friend_timeout_get_id

/* Response structure */
struct gecko_msg_mesh_prov_friend_timeout_get_rsp_t
{
  uint16 result;
};
```

**Table 2.755. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_friend_timeout_status | **Deprecated** and replaced by mesh_config_client_lpn_polltimeout_status event.<br><br>Friend timeout state report. |

#### 2.22.1.14 (deprecated) cmd_mesh_prov_get_config

**Deprecated** and replaced by the following: mesh_config_client_get_beacon, mesh_config_client_get_default_ttl, mesh_config_client_get_friend, mesh_config_client_get_gatt_proxy, mesh_config_client_get_identity, mesh_config_client_get_lpn_polltimeout, and mesh_config_client_get_relay commands.

Get a configuration state value of a node.

Node Configuration Server model state contains a number of node-wide values (for example, node's default TTL value) which are represented as single bytes and can be queried with this command. See the list of configuration states for reference.

Query the more complex states (for example, model-application key bindings) using the commands dedicated for the purpose; see, e.g., get model application key bindings command.

Node response is reported with an configuration status event.

**Table 2.756. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | id | The state to read |
| 8-9 | uint16 | netkey_index | Ignored for node-wide states. |

**Table 2.757. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_get_config_rsp_t *gecko_cmd_mesh_prov_get_config(uint16 address, uint16 id, uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_get_config_id

/* Response structure */
struct gecko_msg_mesh_prov_get_config_rsp_t
{
  uint16 result;
};
```

**Table 2.758. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.15 (deprecated) cmd_mesh_prov_get_dcd

**Deprecated** and replaced by mesh_config_client_get_dcd command.

Get the DCD of the device from a remote Configuration Server. If the call succeeds, the retrieved DCD will be returned in a DCD status event.

**Table 2.759. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6 | uint8 | page | Page number for requested DCD. Use 0xff to get highest existing page. |

**Table 2.760. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_get_dcd_rsp_t *gecko_cmd_mesh_prov_get_dcd(uint16 address, uint8 page);

/* Response id */
gecko_rsp_mesh_prov_get_dcd_id

/* Response structure */
struct gecko_msg_mesh_prov_get_dcd_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.16 (deprecated) cmd_mesh_prov_get_default_configuration_timeout

**Deprecated** and replaced by mesh_config_client_get_default_timeout command.

Get the default timeout for configuration client requests. If there is no response when the timeout expires, a configuration request is considered to have failed.

**Table 2.761. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x33 | method | Message ID |

**Table 2.762. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x33 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | timeout | Timeout in milliseconds. Default timeout is 5 s (5000 ms). |
| 10-13 | uint32 | lpn_timeout | Timeout in milliseconds when communicating with an LPN node. Default LPN timeout is 120 s (120000 ms). |

**BGLIB C API**

```
/* Function */
struct                       gecko_msg_mesh_prov_get_default_configuration_timeout_rsp_t
*gecko_cmd_mesh_prov_get_default_configuration_timeout();

/* Response id */
gecko_rsp_mesh_prov_get_default_configuration_timeout_id

/* Response structure */
struct gecko_msg_mesh_prov_get_default_configuration_timeout_rsp_t
{
  uint16 result;,
  uint32 timeout;,
  uint32 lpn_timeout;
};
```

#### 2.22.1.17 cmd_mesh_prov_get_key_refresh_appkey_blacklist

Check the application key refresh blacklist status of a node. Nodes which are blacklisted for a given application key do not receive updates for that particular application key, but do participate in the key refresh procedure as a whole. This enables the Provisioner to set up and update restricted sets of application keys across the nodes.

**Table 2.763. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x38 | method | Message ID |
| 4-5 | uint16 | netkey_index | Network key index |
| 6-7 | uint16 | appkey_index | Application key index |
| 8 | uint8array | uuid | UUID of the Device |

**Table 2.764. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x38 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | status | Non-zero for blacklisted node |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_prov_get_key_refresh_appkey_blacklist_rsp_t
*gecko_cmd_mesh_prov_get_key_refresh_appkey_blacklist(uint16   netkey_index,   uint16   appkey_index,   uint8
uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_get_key_refresh_appkey_blacklist_id

/* Response structure */
struct gecko_msg_mesh_prov_get_key_refresh_appkey_blacklist_rsp_t
{
  uint16 result;,
  uint8 status;
};
```

#### 2.22.1.18 cmd_mesh_prov_get_key_refresh_blacklist

Check the key refresh blacklist status of a node. Blacklisted nodes do not participate in the key refresh procedure and can therefore be shut out of the network.

**Table 2.765. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | key | Network key index |
| 6 | uint8array | uuid | UUID of the Device |

**Table 2.766. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | status | Non-zero for blacklisted node |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_get_key_refresh_blacklist_rsp_t
*gecko_cmd_mesh_prov_get_key_refresh_blacklist(uint16 key, uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_get_key_refresh_blacklist_id

/* Response structure */
struct gecko_msg_mesh_prov_get_key_refresh_blacklist_rsp_t
{
  uint16 result;,
  uint8 status;
};
```

#### 2.22.1.19 cmd_mesh_prov_get_key_refresh_phase

Get the key refresh phase of an ongoing key refresh procedure.

**Table 2.767. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3e | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key identifying an ongoing key refresh procedure |

**Table 2.768. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | phase | Current key refresh phase. Values are as follows:<br>• 0: Normal operation (no ongoing key refresh)<br>• 1: First phase of key refresh procedure (key deployment)<br>• 2: Second phase of key refresh procedure (new key activation)<br>• 3: Third phase of key refresh procedure (old key deletion) |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_prov_get_key_refresh_phase_rsp_t   *gecko_cmd_mesh_prov_get_key_refresh_phase(uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_get_key_refresh_phase_id

/* Response structure */
struct gecko_msg_mesh_prov_get_key_refresh_phase_rsp_t
{
  uint16 result;,
  uint8 phase;
};
```

#### 2.22.1.20 (deprecated) cmd_mesh_prov_heartbeat_publication_get

**Deprecated** and replaced by mesh_config_client_get_heartbeat_pub command.

Get heartbeat publication state of a node. Node response will be reported as a heartbeat publication status event.

**Table 2.769. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x23 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | Network key index used to encrypt the request |

**Table 2.770. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x23 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_heartbeat_publication_get_rsp_t
*gecko_cmd_mesh_prov_heartbeat_publication_get(uint16 address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_prov_heartbeat_publication_get_id

/* Response structure */
struct gecko_msg_mesh_prov_heartbeat_publication_get_rsp_t
{
  uint16 result;
};
```

**Table 2.771. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_heartbeat_publication_status | **Deprecated** and replaced by mesh_config_client_heartbeat_pub_status event.<br>Node heartbeat status, generated in response to a get heartbeat publication state or set heartbeat publication state request. |

#### 2.22.1.21 (deprecated) cmd_mesh_prov_heartbeat_publication_set

**Deprecated** and replaced by mesh_config_client_set_heartbeat_pub command.

Set heartbeat publication state of a node. Node response will be reported as a heartbeat publication status event.

**Table 2.772. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x24 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request. |
| 8-9 | uint16 | publication_address | Heartbeat publication address. The address cannot be a virtual address. Note that it can be the unassigned address, in which case the heartbeat publishing is disabled. |
| 10 | uint8 | count_log | Heartbeat publication count setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node sends 2^(n-1) heartbeat messages<br>• **0x12 .. 0xfe:** Prohibited<br>• **0xff:** Heartbeat messages are sent indefinitely |
| 11 | uint8 | period_log | Heartbeat publication period setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node sends a heartbeat message every 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |
| 12 | uint8 | ttl | Time-to-live parameter for heartbeat messages |
| 13-14 | uint16 | features | Heartbeat trigger setting. For bits set in the bitmask, reconfiguration of the node feature associated with the bit will result in the node emitting a heartbeat message. Valid values are as follows:<br>• **Bit 0:** Relay feature<br>• **Bit 1:** Proxy feature<br>• **Bit 2:** Friend feature<br>• **Bit 3:** Low power feature<br>Remaining bits are reserved for future use. |
| 15-16 | uint16 | publication_net-key_index | Index of the network key used to encrypt heartbeat messages |

**Table 2.773. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x24 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_heartbeat_publication_set_rsp_t
*gecko_cmd_mesh_prov_heartbeat_publication_set(uint16     address,     uint16     netkey_index,     uint16
publication_address,   uint8   count_log,   uint8   period_log,   uint8   ttl,   uint16   features,   uint16
publication_netkey_index);

/* Response id */
gecko_rsp_mesh_prov_heartbeat_publication_set_id

/* Response structure */
struct gecko_msg_mesh_prov_heartbeat_publication_set_rsp_t
{
  uint16 result;
};
```

**Table 2.774. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_heartbeat_publication_status | **Deprecated** and replaced by mesh_config_client_heartbeat_pub_status event.<br><br>Node heartbeat status, generated in response to a get heartbeat publication state or set heartbeat publication state request. |

#### 2.22.1.22 (deprecated) cmd_mesh_prov_heartbeat_subscription_get

**Deprecated** and replaced by mesh_config_client_get_heartbeat_sub command.

Get node heartbeat subscription state. The node will respond with a subscription status event.

**Table 2.775. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used to encrypt the request |

**Table 2.776. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_heartbeat_subscription_get_rsp_t
*gecko_cmd_mesh_prov_heartbeat_subscription_get(uint16 address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_prov_heartbeat_subscription_get_id

/* Response structure */
struct gecko_msg_mesh_prov_heartbeat_subscription_get_rsp_t
{
  uint16 result;
};
```

**Table 2.777. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_heartbeat_subscription_status | **Deprecated** and replaced by mesh_config_client_heartbeat_sub_status event.<br>Node heartbeat subscription report. |

#### 2.22.1.23 (deprecated) cmd_mesh_prov_heartbeat_subscription_set

**Deprecated** and replaced by mesh_config_client_set_heartbeat_sub command.

Get node heartbeat subscription state. The node will respond with a subscription status event.

**Table 2.778. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x26 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |
| 8-9 | uint16 | subscription_source | Source address for heartbeat messages. Must be either a unicast address or the unassigned address, in which case heartbeat messages are not processed. |
| 10-11 | uint16 | subscription_destination | Destination address for heartbeat messages. The address must be either the unicast address of the primary element of the node, a group address, or the unassigned address. If it is the unassigned address, heartbeat messages are not processed. |
| 12 | uint8 | period_log | Heartbeat subscription period setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not received<br>• **0x01 .. 0x11:** Node receives heartbeat messages for 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |

**Table 2.779. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x26 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_heartbeat_subscription_set_rsp_t
*gecko_cmd_mesh_prov_heartbeat_subscription_set(uint16     address,     uint16     netkey_index,     uint16
subscription_source, uint16 subscription_destination, uint8 period_log);

/* Response id */
gecko_rsp_mesh_prov_heartbeat_subscription_set_id

/* Response structure */
```

```
struct gecko_msg_mesh_prov_heartbeat_subscription_set_rsp_t
{
  uint16 result;
};
```

**Table 2.780. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_heartbeat_subscription_status | **Deprecated** and replaced by mesh_config_client_heartbeat_sub_status event.<br><br>Node heartbeat subscription report. |

#### 2.22.1.24 cmd_mesh_prov_init

Initialize the Bluetooth mesh stack in Provisioner role. When initialization is complete, a provisioner initialized event will be generated.

This command must be issued before any other Bluetooth mesh stack commands. Note that the Bluetooth mesh stack can be initialized either in the Provisioner or the Node role, but not both.

**Table 2.781. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x00 | method | Message ID |

**Table 2.782. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_init_rsp_t *gecko_cmd_mesh_prov_init();

/* Response id */
gecko_rsp_mesh_prov_init_id

/* Response structure */
struct gecko_msg_mesh_prov_init_rsp_t
{
  uint16 result;
};
```

**Table 2.783. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_initialized | Provisioner is initialized and operational. |

#### 2.22.1.25 cmd_mesh_prov_initialize_network

Initialize mesh network and assign provisioner address and IV index for the network. If this command is not invoked prior to invoking mesh prov create network, the network will be initialized with default address and IV index.

**Table 2.784. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x37 | method | Message ID |
| 4-5 | uint16 | address | Address to assign for provisioner. |
| 6-9 | uint32 | ivi | IV index of the network. |

**Table 2.785. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x37 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_initialize_network_rsp_t  *gecko_cmd_mesh_prov_initialize_network(uint16  address,
uint32 ivi);

/* Response id */
gecko_rsp_mesh_prov_initialize_network_id

/* Response structure */
struct gecko_msg_mesh_prov_initialize_network_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.26 cmd_mesh_prov_key_refresh_resume

Resume a suspended key refresh procedure.

By resuming a suspended key refresh procedure the Provisioner will again start to send requests for updating keys or setting key refresh phase to the network.

**Table 2.786. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3d | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key identifying a suspended key refresh procedure |

**Table 2.787. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct     gecko_msg_mesh_prov_key_refresh_resume_rsp_t     *gecko_cmd_mesh_prov_key_refresh_resume(uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_key_refresh_resume_id

/* Response structure */
struct gecko_msg_mesh_prov_key_refresh_resume_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.27 cmd_mesh_prov_key_refresh_start

Start a key refresh procedure in the network.

A key refresh procedure updates a network key and optionally application keys associated with it in all nodes of the network except for blacklisted nodes. After the refresh procedure is complete, the old keys will be discarded. Therefore, the blacklisted nodes, which did not receive new keys will be shut out of the network at the completion of the procedure.

**Table 2.788. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key to update |
| 6 | uint8 | num_appkeys | Number of application keys to update; may be zero. |
| 7 | uint8array | appkey_indices | Indices of the application keys to update, represented as little endian two byte sequences. The array must contain num_appkeys indices and therefore 2*num_appkeys bytes total. |

**Table 2.789. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_key_refresh_start_rsp_t *gecko_cmd_mesh_prov_key_refresh_start(uint16 netkey_index,
uint8 num_appkeys, uint8 appkey_indices_len, const uint8 *appkey_indices_data);

/* Response id */
gecko_rsp_mesh_prov_key_refresh_start_id

/* Response structure */
struct gecko_msg_mesh_prov_key_refresh_start_rsp_t
{
  uint16 result;
};
```

**Table 2.790. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_key_refresh_node_update | Key refresh phase change for a node has occurred. This event is generated when a particular node has moved to a new key refresh phase. |
| mesh_prov_key_refresh_phase_update | Key refresh phase change for a network key has occurred. This event is generated when all nodes participating in a key refresh procedure have been moved to a new state (or have timed out, dropping them from the key refresh procedure). |
| mesh_prov_key_refresh_complete | Key refresh for a network key has completed |

#### 2.22.1.28 cmd_mesh_prov_key_refresh_start_from_phase

Start a key refresh procedure from a non-default phase. Before calling this the keys to be used in the key refresh procedure should have been specified by calling prepare key refresh command.

Note that this command should not normally be used. It is intended only for resuming an interrupted key refresh procedure on a backup Provisioner when the original Provisioner that started the key refresh procedure is no longer available to complete the procedure.

**Table 2.791. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3f | method | Message ID |
| 4 | uint8 | phase | Current key refresh phase |
| 5-6 | uint16 | netkey_index | Index of the network key identifying a key refresh procedure |
| 7 | uint8 | num_appkeys | Number of application keys to update; may be zero. |
| 8 | uint8array | appkey_indices | Indices of the application keys to update, represented as little endian two byte sequences. The array must contain num_appkeys indices and therefore 2*num_appkeys bytes total. |

**Table 2.792. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_key_refresh_start_from_phase_rsp_t
*gecko_cmd_mesh_prov_key_refresh_start_from_phase(uint8 phase, uint16 netkey_index, uint8 num_appkeys, uint8
appkey_indices_len, const uint8 *appkey_indices_data);

/* Response id */
gecko_rsp_mesh_prov_key_refresh_start_from_phase_id

/* Response structure */
struct gecko_msg_mesh_prov_key_refresh_start_from_phase_rsp_t
{
  uint16 result;
};
```

**Table 2.793. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_key_refresh_node_update | Key refresh phase change for a node has occurred. This event is generated when a particular node has moved to a new key refresh phase. |
| mesh_prov_key_refresh_phase_update | Key refresh phase change for a network key has occurred. This event is generated when all nodes participating in a key refresh procedure have been moved to a new state (or have timed out, dropping them from the key refresh procedure). |
| mesh_prov_key_refresh_complete | Key refresh for a network key has completed |

#### 2.22.1.29 cmd_mesh_prov_key_refresh_suspend

Suspend an ongoing key refresh procedure.

Suspending a key refresh procedure means no further requests for updating keys or setting key refresh phase will be sent to the network by the Provisioner until the key refresh procedure is resumed.

**Table 2.794. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3c | method | Message ID |
| 4-5 | uint16 | netkey_index | Index of the network key identifying an ongoing key refresh procedure |

**Table 2.795. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_prov_key_refresh_suspend_rsp_t    *gecko_cmd_mesh_prov_key_refresh_suspend(uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_key_refresh_suspend_id

/* Response structure */
struct gecko_msg_mesh_prov_key_refresh_suspend_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.30 (deprecated) cmd_mesh_prov_model_app_bind

**Deprecated** and replaced by mesh_config_client_bind_model command.

Bind a model to an application key. Node response is reported with a configuration status event.

**Table 2.796. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the configured model |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | appkey_index | The application key to use for binding |
| 12-13 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 14-15 | uint16 | model_id | Model ID of the configured model |

**Table 2.797. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_app_bind_rsp_t  *gecko_cmd_mesh_prov_model_app_bind(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_prov_model_app_bind_id

/* Response structure */
struct gecko_msg_mesh_prov_model_app_bind_rsp_t
{
  uint16 result;
};
```

**Table 2.798. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.31 (deprecated) cmd_mesh_prov_model_app_get

**Deprecated** and replaced by mesh_config_client_list_bindings command.

Get application keys to which the model is bound. Node response is reported with a configuration status event.

**Table 2.799. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the configured model |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of model being configured. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |

**Table 2.800. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_app_get_rsp_t  *gecko_cmd_mesh_prov_model_app_get(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_prov_model_app_get_id

/* Response structure */
struct gecko_msg_mesh_prov_model_app_get_rsp_t
{
  uint16 result;
};
```

**Table 2.801. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.32 (deprecated) cmd_mesh_prov_model_app_unbind

**Deprecated** and replaced by mesh_config_client_unbind_model command.

Remove application key binding from a model. Node response is reported with a configuration status event.

**Table 2.802. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the configured model |
| 8-9 | uint16 | netkey_index | The network key index for encrypting the request |
| 10-11 | uint16 | appkey_index | The index of the application key used in the binding to be removed |
| 12-13 | uint16 | vendor_id | Vendor ID of configured model. Use 0xffff for Bluetooth SIG models |
| 14-15 | uint16 | model_id | Model ID of the configured model. |

**Table 2.803. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_model_app_unbind_rsp_t *gecko_cmd_mesh_prov_model_app_unbind(uint16 address, uint16
elem_address, uint16 netkey_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_prov_model_app_unbind_id

/* Response structure */
struct gecko_msg_mesh_prov_model_app_unbind_rsp_t
{
  uint16 result;
};
```

**Table 2.804. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.33 (deprecated) cmd_mesh_prov_model_pub_get

**Deprecated** and replaced by mesh_config_client_get_model_pub command.

Get a model's publication address, key, and parameters. Node response is reported with a model publication parameters event.

**Table 2.805. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2d | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 10-11 | uint16 | model_id | Model ID of the configured model |

**Table 2.806. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_pub_get_rsp_t  *gecko_cmd_mesh_prov_model_pub_get(uint16  address,  uint16
elem_address, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_prov_model_pub_get_id

/* Response structure */
struct gecko_msg_mesh_prov_model_pub_get_rsp_t
{
  uint16 result;
};
```

**Table 2.807. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_model_pub_status | **Deprecated** and replaced by mesh_config_client_model_pub_status event. |

#### 2.22.1.34 (deprecated) cmd_mesh_prov_model_pub_set

**Deprecated** and replaced by mesh_config_client_set_model_pub command.

Set a model's publication address, key, and parameters. Node response is reported with a configuration status event.

**Table 2.808. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the configured model |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | appkey_index | The application key index to use for the published messages |
| 12-13 | uint16 | vendor_id | Vendor ID of model being configured. Use 0xffff for Bluetooth SIG models. |
| 14-15 | uint16 | model_id | Model ID of the configured model. |
| 16-17 | uint16 | pub_address | The address to publish to. It can be a unicast address, a virtual address, or a group address. It can also be the unassigned address to stop the model from publishing. |
| 18 | uint8 | ttl | Publication time-to-live value |
| 19 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 20 | uint8 | retrans | Retransmission count and interval, which controls how many times the model re-publishes the same message after the initial publish transmission and the cadence of retransmissions.<br>Retransmission count is encoded in the three low bits of the value, ranging from 0 to 7. Default value is 0 (no retransmissions).<br>Retransmission interval is encoded in the five high bits of the value, ranging from 0 to 31, in 50-millisecond units. Value of 0 corresponds to 50 ms, while value of 31 corresponds to 1600 ms. |

**Table 2.809. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |

| Byte | Type | Name | Description |
|---|---|---|---|
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_pub_set_rsp_t  *gecko_cmd_mesh_prov_model_pub_set(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id, uint16 pub_address,
uint8 ttl, uint8 period, uint8 retrans);

/* Response id */
gecko_rsp_mesh_prov_model_pub_set_id

/* Response structure */
struct gecko_msg_mesh_prov_model_pub_set_rsp_t
{
  uint16 result;
};
```

**Table 2.810. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.35 (deprecated) cmd_mesh_prov_model_pub_set_cred

**Deprecated** and replaced by mesh_config_client_set_model_pub command.

This command is otherwise the same as the regular model publication set command but it also has a parameter for setting the Friendship Credential Flag.

**Table 2.811. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2f | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index for encrypting the request |
| 10-11 | uint16 | appkey_index | The application key index for the published messages |
| 12-13 | uint16 | vendor_id | Vendor ID of the model being configured. Use 0xffff for Bluetooth SIG models. |
| 14-15 | uint16 | model_id | Model ID of the configured model |
| 16-17 | uint16 | pub_address | The address to publish to. It can be a unicast address, a virtual address, or a group address. It can also be the unassigned address to stop the model from publishing. |
| 18 | uint8 | ttl | Publication time-to-live value |
| 19 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 20 | uint8 | retrans | See documentation of model publication set command for details. |
| 21 | uint8 | credentials | Friendship credential flag. If the value is zero, publication is done using normal credentials. If the value is one, it is done with friendship credentials, meaning only the friend can decrypt the published message and relay it forward using the normal credentials. The default value is 0. |

**Table 2.812. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |

| Byte | Type | Name | Description |
|---|---|---|---|
| 3 | 0x2f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_pub_set_cred_rsp_t  *gecko_cmd_mesh_prov_model_pub_set_cred(uint16  address,
uint16  elem_address,  uint16  netkey_index,  uint16  appkey_index,  uint16  vendor_id,  uint16  model_id,  uint16
pub_address, uint8 ttl, uint8 period, uint8 retrans, uint8 credentials);

/* Response id */
gecko_rsp_mesh_prov_model_pub_set_cred_id

/* Response structure */
struct gecko_msg_mesh_prov_model_pub_set_cred_rsp_t
{
  uint16 result;
};
```

**Table 2.813. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.36 (deprecated) cmd_mesh_prov_model_pub_set_va

**Deprecated** and replaced by mesh_config_client_set_model_pub_va command.

Set a model's publication virtual address, key, and parameters. Node response is reported with a configuration status event.

**Table 2.814. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2e | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured. |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | appkey_index | The application key index to use for the published messages |
| 12-13 | uint16 | vendor_id | Vendor ID of the model being configured. Use 0xffff for Bluetooth SIG models. |
| 14-15 | uint16 | model_id | Model ID of the configured model |
| 16 | uint8 | ttl | Publication time-to-live value |
| 17 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 18 | uint8 | retrans | See documentation of model publication set command for details. |
| 19 | uint8array | pub_address | The Label UUID to publish to. The byte array must be exactly 16 bytes long. |

**Table 2.815. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_model_pub_set_va_rsp_t *gecko_cmd_mesh_prov_model_pub_set_va(uint16 address, uint16
elem_address, uint16 netkey_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id, uint8 ttl, uint8
period, uint8 retrans, uint8 pub_address_len, const uint8 *pub_address_data);

/* Response id */
gecko_rsp_mesh_prov_model_pub_set_va_id

/* Response structure */
struct gecko_msg_mesh_prov_model_pub_set_va_rsp_t
{
  uint16 result;
};
```

**Table 2.816. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.37 (deprecated) cmd_mesh_prov_model_pub_set_va_cred

**Deprecated** and replaced by mesh_config_client_set_model_pub_va command.

This command is otherwise the same as the regular model publication set virtual address command but it also has a parameter for setting the Friendship Credential Flag.

**Table 2.817. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x30 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index for encrypting the request. |
| 10-11 | uint16 | appkey_index | The application key index for the published messages. |
| 12-13 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 14-15 | uint16 | model_id | Model ID of the configured model |
| 16 | uint8 | ttl | Publication time-to-live value |
| 17 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 18 | uint8 | retrans | See documentation of model publication set command for details. |
| 19 | uint8 | credentials | Friendship credential flag. If the value is zero, publication is done using normal credentials. If the value is one, it is done with friendship credentials, meaning only the friend can decrypt the published message and relay it forward using the normal credentials. The default value is 0. |
| 20 | uint8array | pub_address | The Label UUID to publish to. The byte array must be exactly 16 bytes long. |

**Table 2.818. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x30 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_pub_set_va_cred_rsp_t  *gecko_cmd_mesh_prov_model_pub_set_va_cred(uint16
address, uint16 elem_address, uint16 netkey_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id,
uint8  ttl,  uint8  period,  uint8  retrans,  uint8  credentials,  uint8  pub_address_len,  const  uint8
*pub_address_data);

/* Response id */
gecko_rsp_mesh_prov_model_pub_set_va_cred_id

/* Response structure */
struct gecko_msg_mesh_prov_model_pub_set_va_cred_rsp_t
{
  uint16 result;
};
```

**Table 2.819. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.38 (deprecated) cmd_mesh_prov_model_sub_add

**Deprecated** and replaced by mesh_config_client_add_model_sub command.

Add an address to a model's subscription list. Node response is reported with a configuration status event.

**Table 2.820. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model to be configured |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |
| 14-15 | uint16 | sub_address | The address to add to the subscription list. Note that the address has to be a group address. |

**Table 2.821. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_sub_add_rsp_t  *gecko_cmd_mesh_prov_model_sub_add(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id, uint16 sub_address);

/* Response id */
gecko_rsp_mesh_prov_model_sub_add_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_add_rsp_t
{
  uint16 result;
};
```

**Table 2.822. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.39 (deprecated) cmd_mesh_prov_model_sub_add_va

**Deprecated** and replaced by mesh_config_client_add_model_sub_va command.

Add an virtual address to a model's subscription list. Node response is reported with a configuration status event.

**Table 2.823. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1f | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request. |
| 10-11 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |
| 14 | uint8array | sub_address | The Label UUID to add to the subscription list. The array must be exactly 16 bytes long. |

**Table 2.824. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_model_sub_add_va_rsp_t *gecko_cmd_mesh_prov_model_sub_add_va(uint16 address, uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id, uint8 sub_address_len, const uint8
*sub_address_data);

/* Response id */
gecko_rsp_mesh_prov_model_sub_add_va_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_add_va_rsp_t
{
  uint16 result;
};
```

**Table 2.825. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.40 (deprecated) cmd_mesh_prov_model_sub_clear

**Deprecated** and replaced by mesh_config_client_clear_model_sub command.

Clear all addresses from a model's subscription list. Node response is reported with a configuration status event.

**Table 2.826. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2c | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |

**Table 2.827. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_model_sub_clear_rsp_t *gecko_cmd_mesh_prov_model_sub_clear(uint16 address, uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_prov_model_sub_clear_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_clear_rsp_t
{
  uint16 result;
};
```

**Table 2.828. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.41 (deprecated) cmd_mesh_prov_model_sub_del

**Deprecated** and replaced by is mesh_config_client_remove_model_sub command.

Remove an address from a model's subscription list. Node response is reported with a configuration status event.

**Table 2.829. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1e | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured. |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |
| 14-15 | uint16 | sub_address | The address to remove from the subscription list |

**Table 2.830. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_sub_del_rsp_t  *gecko_cmd_mesh_prov_model_sub_del(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id, uint16 sub_address);

/* Response id */
gecko_rsp_mesh_prov_model_sub_del_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_del_rsp_t
{
  uint16 result;
};
```

**Table 2.831. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.42 (deprecated) cmd_mesh_prov_model_sub_del_va

**Deprecated** and replaced by mesh_config_client_remove_model_sub_va command.

Remove a virtual address from a model's subscription list. Node response is reported with a configuration status event.

**Table 2.832. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x20 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |
| 14 | uint8array | sub_address | The Label UUID to remove from the subscription list. The array must be exactly 16 bytes long. |

**Table 2.833. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x20 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_model_sub_del_va_rsp_t *gecko_cmd_mesh_prov_model_sub_del_va(uint16 address, uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id, uint8 sub_address_len, const uint8
*sub_address_data);

/* Response id */
gecko_rsp_mesh_prov_model_sub_del_va_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_del_va_rsp_t
{
  uint16 result;
};
```

**Table 2.834. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.43 (deprecated) cmd_mesh_prov_model_sub_get

**Deprecated** and replaced by mesh_config_client_list_subs command.

Get a model's subscription list. Node response is reported with subscription list entry and events.

**Table 2.835. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x31 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the model being configured. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |

**Table 2.836. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x31 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_sub_get_rsp_t  *gecko_cmd_mesh_prov_model_sub_get(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_prov_model_sub_get_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_get_rsp_t
{
  uint16 result;
};
```

**Table 2.837. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_model_sub_addr | **Deprecated** and replaced by mesh_config_client_subs_list event.<br><br>This event is generated once for each subscription address a model reports when its subscription list is queried using the get subscription list command. The list is terminated with the subscription list entries end event. |
| mesh_prov_model_sub_addr_end | **Deprecated** and replace by mesh_config_client_subs_list_end event.<br><br>This event terminates model subscription list result reporting. |

### 2.22.1.44 (deprecated) cmd_mesh_prov_model_sub_set

**Deprecated** and replaced by mesh_config_client_set_model_sub command.

Set an address to a model's subscription list, overwriting previous contents. Node response is reported with a configuration status event.

**Table 2.838. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x21 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the model being configured. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |
| 14-15 | uint16 | sub_address | The address to set as the subscription list. Note that the address has to be a group address. |

**Table 2.839. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x21 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_model_sub_set_rsp_t  *gecko_cmd_mesh_prov_model_sub_set(uint16  address,  uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id, uint16 sub_address);

/* Response id */
gecko_rsp_mesh_prov_model_sub_set_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_set_rsp_t
{
  uint16 result;
};
```

**Table 2.840. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

### 2.22.1.45 (deprecated) cmd_mesh_prov_model_sub_set_va

**Deprecated** and replaced by mesh_config_client_set_model_sub_va command.

Set a virtual address to a model's subscription list overwriting previous content. Node response is reported with a configuration status event.

**Table 2.841. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x22 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the model, which will be configured |
| 8-9 | uint16 | netkey_index | The network key index used for encrypting the request |
| 10-11 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 12-13 | uint16 | model_id | Model ID of the configured model |
| 14 | uint8array | sub_address | The Label UUID to set as the subscription list. The byte array must be exactly 16 bytes long. |

**Table 2.842. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x22 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_model_sub_set_va_rsp_t *gecko_cmd_mesh_prov_model_sub_set_va(uint16 address, uint16
elem_address, uint16 netkey_index, uint16 vendor_id, uint16 model_id, uint8 sub_address_len, const uint8
*sub_address_data);

/* Response id */
gecko_rsp_mesh_prov_model_sub_set_va_id

/* Response structure */
struct gecko_msg_mesh_prov_model_sub_set_va_rsp_t
{
  uint16 result;
};
```

**Table 2.843. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.46 (deprecated) cmd_mesh_prov_nettx_get

**Deprecated** and replaced by mesh_config_client_get_network_transmit command.

Retrieve network layer transmission parameters of a node.

**Table 2.844. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1c | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |

**Table 2.845. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_nettx_get_rsp_t *gecko_cmd_mesh_prov_nettx_get(uint16 address);

/* Response id */
gecko_rsp_mesh_prov_nettx_get_id

/* Response structure */
struct gecko_msg_mesh_prov_nettx_get_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.47 (deprecated) cmd_mesh_prov_nettx_set

**Deprecated** and replaced by mesh_config_client_set_network_transmit command.

Set network layer transmission parameters of a node.

**Table 2.846. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1d | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6 | uint8 | count | Retransmission count (excluding initial transmission). Range: 0..7; the default value is 0 (no retransmissions). |
| 7 | uint8 | interval | Retransmission interval in 10-millisecond steps |

**Table 2.847. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_nettx_set_rsp_t *gecko_cmd_mesh_prov_nettx_set(uint16 address, uint8 count, uint8
interval);

/* Response id */
gecko_rsp_mesh_prov_nettx_set_id

/* Response structure */
struct gecko_msg_mesh_prov_nettx_set_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.48 (deprecated) cmd_mesh_prov_network_add

**Deprecated** and replaced by mesh_config_client_add_netkey command.

Push a network key to a node. The key must exist on the Provisioner (see create network key command).

Node response is reported with an configuration status event.

**Table 2.848. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | netkey_index | The index of the key to push to the node |

**Table 2.849. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_prov_network_add_rsp_t   *gecko_cmd_mesh_prov_network_add(uint16   address,   uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_network_add_id

/* Response structure */
struct gecko_msg_mesh_prov_network_add_rsp_t
{
  uint16 result;
};
```

**Table 2.850. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.49 (deprecated) cmd_mesh_prov_network_delete

**Deprecated** and replaced by mesh_config_client_remove_netkey command.

Delete a network key on a node.

When a network key is deleted, the application keys bound to it are deleted automatically. There is no need to explicitly use the delete application key command.

Note that it is not possible to delete the key used in encrypting the command itself (which is the first network key deployed to the node during provisioning), otherwise the node would not be able to respond.

Node response is reported with an configuration status event.

**Table 2.851. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | netkey_index | The index of the key to delete |

**Table 2.852. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_network_delete_rsp_t  *gecko_cmd_mesh_prov_network_delete(uint16  address,  uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_network_delete_id

/* Response structure */
struct gecko_msg_mesh_prov_network_delete_rsp_t
{
  uint16 result;
};
```

**Table 2.853. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.50 (deprecated) cmd_mesh_prov_network_get

**Deprecated** and replaced by mesh_config_client_list_netkeys command.

Get a list of network keys bound from a node.

Return a list of network key indices of network keys deployed to a node.

Node response is reported with a number of network key listevents, terminated by a event.

**Table 2.854. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2b | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |

**Table 2.855. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x2b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_network_get_rsp_t *gecko_cmd_mesh_prov_network_get(uint16 address);

/* Response id */
gecko_rsp_mesh_prov_network_get_id

/* Response structure */
struct gecko_msg_mesh_prov_network_get_rsp_t
{
  uint16 result;
};
```

**Table 2.856. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_network_list | **Deprecated** and replaced by mesh_config_client_netkey_list event.<br>Network key list event. |

| Event | Description |
|---|---|
| mesh_prov_network_list_end | **Deprecated** and replaced by mesh_config_client_netkey_list_end event.<br>Network key list terminator event. |

#### 2.22.1.51 cmd_mesh_prov_oob_auth_rsp

Respond to prov_oob_auth_request

**Table 2.857. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x09 | method | Message ID |
| 4 | uint8array | data | Output or static OOB data |

**Table 2.858. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_oob_auth_rsp_rsp_t *gecko_cmd_mesh_prov_oob_auth_rsp(uint8 data_len, const uint8
*data_data);

/* Response id */
gecko_rsp_mesh_prov_oob_auth_rsp_id

/* Response structure */
struct gecko_msg_mesh_prov_oob_auth_rsp_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.52 cmd_mesh_prov_oob_pkey_rsp

Respond to prov_oob_pkey_request

**Table 2.859. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x08 | method | Message ID |
| 4 | uint8array | pkey | Public Key read out-of-band |

**Table 2.860. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_oob_pkey_rsp_rsp_t *gecko_cmd_mesh_prov_oob_pkey_rsp(uint8 pkey_len, const uint8
*pkey_data);

/* Response id */
gecko_rsp_mesh_prov_oob_pkey_rsp_id

/* Response structure */
struct gecko_msg_mesh_prov_oob_pkey_rsp_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.53 cmd_mesh_prov_provision_device

Provision a device into a network using the advertisement bearer (PB-ADV).

Issuing this command starts the provisioning process for the specified device. After the process completes successfully, a device provisioned event is generated. If provisioning fails, a provisioning failed event will be generated instead.

**Table 2.861. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | network_id | Index of the initial network key, which is sent to the device during provisioning |
| 5 | uint8array | uuid | UUID of the device to provision |

**Table 2.862. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_prov_provision_device_rsp_t  *gecko_cmd_mesh_prov_provision_device(uint8  network_id,
uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_provision_device_id

/* Response structure */
struct gecko_msg_mesh_prov_provision_device_rsp_t
{
  uint16 result;
};
```

**Table 2.863. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_device_provisioned | Device provisioned successfully. |
| mesh_prov_provisioning_failed | Provisioning a device failed. |

#### 2.22.1.54 cmd_mesh_prov_provision_device_with_address

Provision a device into a network using the advertisement bearer (PB-ADV). Application may specify the unicast addresses given for the device elements with this command.

Issuing this command starts the provisioning process for the specified device. Once the process completes successfully, a device provisioned event is generated. If provisioning does not succeed, a provisioning failed event will be generated instead.

**Table 2.864. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x35 | method | Message ID |
| 4 | uint8 | network_id | Index of the initial network key, which is sent to the device during provisioning |
| 5-6 | uint16 | address | Unicast address for the primary element of the device. If unassigned address (0x0000) is given, the stack will automatically assign an address. Note that element addresses are contiguous and must all be unicast addresses; primary address must be chosen so that also the last element's address will still be a unicast address. |
| 7 | uint8 | elements | Maximum number of elements in device to be provisioned. If the primary element address is set to unassigned, this needs to be set to zero. |
| 8 | uint8 | attention_timer | Attention timer value, in seconds, which indicates the time that the provisioned device should attract human attention |
| 9 | uint8array | uuid | UUID of the device to provision |

**Table 2.865. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x35 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_provision_device_with_address_rsp_t
*gecko_cmd_mesh_prov_provision_device_with_address(uint8  network_id,  uint16  address,  uint8  elements,  uint8
attention_timer, uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_provision_device_with_address_id
```

```
/* Response structure */
struct gecko_msg_mesh_prov_provision_device_with_address_rsp_t
{
  uint16 result;
};
```

**Table 2.866. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_device_provisioned | Device provisioned successfully. |
| mesh_prov_provisioning_failed | Provisioning a device failed. |

#### 2.22.1.55 cmd_mesh_prov_provision_gatt_device

Provision a device into a network using the GATT bearer (PB-GATT).

Issuing this command starts the provisioning process for the specified device. After the process completes successfully, a device_provisioned event is generated. If provisioning fails, a provisioning failed event is generated instead.

Note that this command is available only if GATT functionality is compiled in to the firmware. If that is not the case, the command will return with a "not implemented" return code.

**Table 2.867. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x15 | method | Message ID |
| 4 | uint8 | network_id | Index of the initial network key, which is sent to the device during provisioning. |
| 5 | uint8 | connection | Connection handle for the device to be provisioned |
| 6 | uint8array | uuid | UUID of the Device to provision |

**Table 2.868. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_prov_provision_gatt_device_rsp_t   *gecko_cmd_mesh_prov_provision_gatt_device(uint8
network_id, uint8 connection, uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_provision_gatt_device_id

/* Response structure */
struct gecko_msg_mesh_prov_provision_gatt_device_rsp_t
{
  uint16 result;
};
```

**Table 2.869. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_device_provisioned | Device provisioned successfully. |
| mesh_prov_provisioning_failed | Provisioning a device failed. |

#### 2.22.1.56 cmd_mesh_prov_provision_gatt_device_with_address

Provision a device into a network using the GATT bearer (PB-GATT). Application may specify the unicast addresses given for the device elements with this command.

Issuing this command starts the provisioning process for the specified device. Once the process completes successfully, a device provisioned event is generated. If provisioning does not succeed, a provisioning failed event will be generated instead.

Note that this command is available only if GATT functionality is compiled in to the firmware. If that is not the case, the command will return with a "not implemented" return code.

**Table 2.870. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x36 | method | Message ID |
| 4 | uint8 | network_id | Index of the initial network key, which is sent to the device during provisioning |
| 5 | uint8 | connection | Connection handle for the device to be provisioned |
| 6-7 | uint16 | address | Unicast address for the primary element of the device. If unassigned address (0x0000) is given, the stack will automatically assign an address. Note that element addresses are contiguous and must all be unicast addresses; primary address must be chosen so that also the last element's address will still be a unicast address. |
| 8 | uint8 | elements | Maximum number of elements in device to be provisioned. If the primary element address is set to unassigned, this needs to be set to zero. |
| 9 | uint8 | attention_timer | Attention timer value, in seconds, indicates the time that the provisioned device should attract human attention |
| 10 | uint8array | uuid | UUID of the device to provision |

**Table 2.871. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x36 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_prov_provision_gatt_device_with_address_rsp_t
*gecko_cmd_mesh_prov_provision_gatt_device_with_address(uint8  network_id,  uint8  connection,  uint16  address,
```

```
uint8 elements, uint8 attention_timer, uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_provision_gatt_device_with_address_id

/* Response structure */
struct gecko_msg_mesh_prov_provision_gatt_device_with_address_rsp_t
{
  uint16 result;
};
```

**Table 2.872. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_device_provisioned | Device provisioned successfully. |
| mesh_prov_provisioning_failed | Provisioning a device failed. |

#### 2.22.1.57 (deprecated) cmd_mesh_prov_relay_get

**Deprecated** and replaced by mesh_config_client_get_relay command.

Get node relay retransmission state. The node will respond with a subscription status event.

**Table 2.873. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x27 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |

**Table 2.874. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x27 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_relay_get_rsp_t *gecko_cmd_mesh_prov_relay_get(uint16 address, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_prov_relay_get_id

/* Response structure */
struct gecko_msg_mesh_prov_relay_get_rsp_t
{
  uint16 result;
};
```

**Table 2.875. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_relay_status | **Deprecated** and replaced by mesh_config_client_relay_status event.<br>Node relay state report. |

#### 2.22.1.58 (deprecated) cmd_mesh_prov_relay_set

**Deprecated** and replaced by mesh_config_client_set_relay command.

Set node relay retransmission state. The node will respond with a relay status event.

**Table 2.876. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x28 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request. |
| 8 | uint8 | relay | Relay state. Valid values are as follows:<br>• **0x00:** Relaying disabled<br>• **0x01:** Relaying enabled |
| 9 | uint8 | count | Relay retransmit count. Value must be between 0 and 7; default value is 0 (no retransmissions). |
| 10 | uint8 | interval | Relay retransmit interval in milliseconds. Value must be between 0 and 31; it represents 10-millisecond increments, starting at 10 ms. |

**Table 2.877. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x28 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_relay_set_rsp_t *gecko_cmd_mesh_prov_relay_set(uint16 address, uint16 netkey_index,
uint8 relay, uint8 count, uint8 interval);

/* Response id */
gecko_rsp_mesh_prov_relay_set_id

/* Response structure */
struct gecko_msg_mesh_prov_relay_set_rsp_t
{
  uint16 result;
};
```

**Table 2.878. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_relay_status | **Deprecated** and replaced by mesh_config_client_relay_status event.<br>Node relay state report. |

#### 2.22.1.59 (deprecated) cmd_mesh_prov_reset_node

**Deprecated** and replaced by mesh_config_client_reset_node command.

Send a reset request to a node.

If a node replies to the request, a node reset event will be generated. Note that the reply packet may get lost and the node has reset itself even in the absence of the event.

Also note that for securely removing a node from the network, perform a key refresh with the removed node blacklisted.

**Table 2.879. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x29 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |

**Table 2.880. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x29 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_prov_reset_node_rsp_t   *gecko_cmd_mesh_prov_reset_node(uint16   address,   uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_prov_reset_node_id

/* Response structure */
struct gecko_msg_mesh_prov_reset_node_rsp_t
{
  uint16 result;
};
```

**Table 2.881. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_node_reset | **Deprecated** and replaced by mesh_config_client_reset_status event.<br><br>A node has reset itself. |

#### 2.22.1.60 cmd_mesh_prov_scan_unprov_beacons

Start scanning for unprovisioned device beacons.

Unprovisioned devices send out beacons containing their UUID. An unprovisioned beacon event will be generated for each beacon seen. Once the UUID of a device is known, the Provisioner may start provisioning the device by issuing either the provision device over PB-ADV or provision device over PB-GATT command.

**Table 2.882. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x01 | method | Message ID |

**Table 2.883. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_scan_unprov_beacons_rsp_t *gecko_cmd_mesh_prov_scan_unprov_beacons();

/* Response id */
gecko_rsp_mesh_prov_scan_unprov_beacons_id

/* Response structure */
struct gecko_msg_mesh_prov_scan_unprov_beacons_rsp_t
{
  uint16 result;
};
```

**Table 2.884. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_unprov_beacon | Unprovisioned beacon seen. |

#### 2.22.1.61 (deprecated) cmd_mesh_prov_set_config

**Deprecated** and replaced by the following: mesh_config_client_set_beacon, mesh_config_client_set_default_ttl, mesh_config_client_set_friend, mesh_config_client_set_gatt_proxy, mesh_config_client_set_identity, and mesh_config_client_set_relay commands.

Set a configuration state value of a node.

Node Configuration Server model state contains a number of node-wide values (for example, node's default TTL value) which are represented as single bytes and can be modified with this command. See the list of configuration states for reference.

Set the more complex states using the commands dedicated for the purpose because this command accepts only raw binary data as the value to set.

Node response is reported with an configuration status event.

**Table 2.885. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node's primary element |
| 6-7 | uint16 | id | State to manipulate |
| 8-9 | uint16 | netkey_index | Ignored for node-wide states. |
| 10 | uint8array | value | Raw binary value |

**Table 2.886. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_set_config_rsp_t *gecko_cmd_mesh_prov_set_config(uint16 address, uint16 id, uint16
netkey_index, uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_mesh_prov_set_config_id

/* Response structure */
struct gecko_msg_mesh_prov_set_config_rsp_t
{
  uint16 result;
};
```

**Table 2.887. Events Generated**

| Event | Description |
|---|---|
| mesh_prov_config_status | **Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.<br><br>Remote status response to a get/set request. |

#### 2.22.1.62 (deprecated) cmd_mesh_prov_set_default_configuration_timeout

**Deprecated** and replaced by mesh_config_client_set_default_timeout command.

Set the default timeout for configuration client requests. If there is no response when the timeout expires, a configuration request is considered to have failed.

**Table 2.888. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x34 | method | Message ID |
| 4-7 | uint32 | timeout | Timeout in milliseconds. Default timeout is 5 s (5000 ms). |
| 8-11 | uint32 | lpn_timeout | Timeout in milliseconds when communicating with an LPN node. Default LPN timeout is 120 s (120000 ms). |

**Table 2.889. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x34 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                       gecko_msg_mesh_prov_set_default_configuration_timeout_rsp_t
*gecko_cmd_mesh_prov_set_default_configuration_timeout(uint32 timeout, uint32 lpn_timeout);

/* Response id */
gecko_rsp_mesh_prov_set_default_configuration_timeout_id

/* Response structure */
struct gecko_msg_mesh_prov_set_default_configuration_timeout_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.63 cmd_mesh_prov_set_key_refresh_appkey_blacklist

Set the application key refresh blacklist status of a node. Nodes which are blacklisted for a given application key do not receive updates for that particular application key, but do participate in the key refresh procedure as a whole. This enables the Provisioner to set up and update restricted sets of application keys across the nodes.

**Table 2.890. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x39 | method | Message ID |
| 4-5 | uint16 | netkey_index | Network key index |
| 6-7 | uint16 | appkey_index | Application key index |
| 8 | uint8 | status | Non-zero for blacklisted node |
| 9 | uint8array | uuid | UUID of the device |

**Table 2.891. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x39 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_prov_set_key_refresh_appkey_blacklist_rsp_t
*gecko_cmd_mesh_prov_set_key_refresh_appkey_blacklist(uint16 netkey_index, uint16 appkey_index, uint8 status,
uint8 uuid_len, const uint8 *uuid_data);

/* Response id */
gecko_rsp_mesh_prov_set_key_refresh_appkey_blacklist_id

/* Response structure */
struct gecko_msg_mesh_prov_set_key_refresh_appkey_blacklist_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.64 cmd_mesh_prov_set_key_refresh_blacklist

Set the key refresh blacklist status of a node. Blacklisted nodes do not participate in the key refresh procedure and can therefore be shut out of the network.

**Table 2.892. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | key | Network key index |
| 6 | uint8 | status | Non-zero for blacklisted node |
| 7 | uint8array | uuid | UUID of the Device |

**Table 2.893. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_prov_set_key_refresh_blacklist_rsp_t
*gecko_cmd_mesh_prov_set_key_refresh_blacklist(uint16  key,  uint8  status,  uint8  uuid_len,  const  uint8
*uuid_data);

/* Response id */
gecko_rsp_mesh_prov_set_key_refresh_blacklist_id

/* Response structure */
struct gecko_msg_mesh_prov_set_key_refresh_blacklist_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.65 cmd_mesh_prov_set_oob_requirements

Set the OOB requirements for devices to be provisioned.

**Table 2.894. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0a | method | Message ID |
| 4 | uint8 | public_key | The public key. Set to zero if the provisioning does not use OOB public Key. |
| 5 | uint8 | auth_methods | Allowed OOB authentication methods The value is a bitmap so that multiple methods can be supported. |
| 6-7 | uint16 | output_actions | Allowed OOB Output Action types |
| 8-9 | uint16 | input_actions | Allowed OOB Input Action types |
| 10 | uint8 | min_size | Minimum input/output OOB size. Values range from 0 (input/output OOB not used) to 8. |
| 11 | uint8 | max_size | Maximum input/output OOB size. Must be smaller than or equal to the minimum size. Values range from 0 (input/output OOB not used) to 8. |

**Table 2.895. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_prov_set_oob_requirements_rsp_t    *gecko_cmd_mesh_prov_set_oob_requirements(uint8
public_key, uint8 auth_methods, uint16 output_actions, uint16 input_actions, uint8 min_size, uint8 max_size);

/* Response id */
gecko_rsp_mesh_prov_set_oob_requirements_id

/* Response structure */
struct gecko_msg_mesh_prov_set_oob_requirements_rsp_t
{
  uint16 result;
};
```

#### 2.22.1.66 cmd_mesh_prov_stop_scan_unprov_beacons

Stop scanning for unprovisioned device beacons.

**Table 2.896. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3a | method | Message ID |

**Table 2.897. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x3a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_prov_stop_scan_unprov_beacons_rsp_t *gecko_cmd_mesh_prov_stop_scan_unprov_beacons();

/* Response id */
gecko_rsp_mesh_prov_stop_scan_unprov_beacons_id

/* Response structure */
struct gecko_msg_mesh_prov_stop_scan_unprov_beacons_rsp_t
{
  uint16 result;
};
```

### 2.22.2 mesh_prov events

#### 2.22.2.1 (deprecated) evt_mesh_prov_appkey_list

**Deprecated** and replaced by mesh_config_client_appkey_list event.

Application key list event.

**Table 2.898. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the node |
| 6-7 | uint16 | netkey_index | Index of the network key to which the listed application key is bound on the node |
| 8-9 | uint16 | appkey_index | Index of the application key |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_appkey_list_id

/* Event structure */
struct gecko_msg_mesh_prov_appkey_list_evt_t
{
  uint16 address;,
  uint16 netkey_index;,
  uint16 appkey_index;
};
```

#### 2.22.2.2 (deprecated) evt_mesh_prov_appkey_list_end

**Deprecated** and replaced by mesh_config_client_appkey_list_end event.

Application key list terminator event.

**Table 2.899. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | address | Unicast address of the node |
| 8-9 | uint16 | netkey_index | Index of the network key to which the listed application key is bound on the node |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_appkey_list_end_id

/* Event structure */
struct gecko_msg_mesh_prov_appkey_list_end_evt_t
{
  uint16 result;,
  uint16 address;,
  uint16 netkey_index;
};
```

#### 2.22.2.3 (deprecated) evt_mesh_prov_config_status

**Deprecated** and replaced by the following: mesh_config_client_beacon_status, mesh_config_client_default_ttl_status, mesh_config_client_friend_status, mesh_config_client_gatt_proxy_status, mesh_config_client_identity_status, mesh_config_client_lpn_polltimeout_status, and mesh_config_client_relay_status events.

Remote status response to a get/set request.

**Table 2.900. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the responder's primary element |
| 6-7 | uint16 | id | The state requested/indicated |
| 8 | uint8 | status | Status code. If non-zero, ignore the data field. |
| 9 | uint8array | data | Raw binary format data |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_config_status_id

/* Event structure */
struct gecko_msg_mesh_prov_config_status_evt_t
{
  uint16 address;,
  uint16 id;,
  uint8 status;,
  uint8array data;
};
```

#### 2.22.2.4 (deprecated) evt_mesh_prov_dcd_status

**Deprecated** and replaced by mesh_config_client_dcd_data event.

This event carries the device composition data of a node.

**Table 2.901. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | address | Unicast address of the target node's primary element |
| 8-9 | uint16 | cid | Company Identifier |
| 10-11 | uint16 | pid | Product Identifier |
| 12-13 | uint16 | vid | Version Identifier |
| 14-15 | uint16 | crpl | Capacity of Replay Protection List |
| 16-17 | uint16 | features | Features bitmask |
| 18 | uint8 | elements | Number of Elements |
| 19 | uint8 | models | Number of models in all of the elements combined |
| 20 | uint8array | element_data | Element Data. Format: [ Location (uint16), SIG Model Count (uint8), Vendor Model Count (uint8), [ SIG Models (uint16) ], [ Vendor Models (uint32) ] ] |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_dcd_status_id

/* Event structure */
struct gecko_msg_mesh_prov_dcd_status_evt_t
{
  uint16 result;,
  uint16 address;,
  uint16 cid;,
  uint16 pid;,
  uint16 vid;,
  uint16 crpl;,
  uint16 features;,
  uint8 elements;,
  uint8 models;,
  uint8array element_data;
};
```

#### 2.22.2.5 evt_mesh_prov_ddb_list

Provisioner's device database list entry

**Table 2.902. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x13 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x09 | method | Message ID |
| 4-19 | uuid_128 | uuid | UUID of the Device |
| 20-21 | uint16 | address | Unicast address of the primary element of the node |
| 22 | uint8 | elements | Number of elements the device has |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_ddb_list_id

/* Event structure */
struct gecko_msg_mesh_prov_ddb_list_evt_t
{
  uuid_128 uuid;,
  uint16 address;,
  uint8 elements;
};
```

#### 2.22.2.6 evt_mesh_prov_device_provisioned

Device provisioned successfully.

**Table 2.903. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | address | Address assigned to the node's primary element. If the node has multiple elements, they have been assigned an address in a consecutive sequence following the primary element address. |
| 6 | uint8array | uuid | UUID of the device |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_device_provisioned_id

/* Event structure */
struct gecko_msg_mesh_prov_device_provisioned_evt_t
{
  uint16 address;,
  uint8array uuid;
};
```

#### 2.22.2.7 (deprecated) evt_mesh_prov_friend_timeout_status

**Deprecated** and replaced by mesh_config_client_lpn_polltimeout_status event.

Friend timeout state report.

**Table 2.904. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |
| 8-11 | uint32 | timeout | |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_friend_timeout_status_id

/* Event structure */
struct gecko_msg_mesh_prov_friend_timeout_status_evt_t
{
  uint16 address;,
  uint16 netkey_index;,
  uint32 timeout;
};
```

#### 2.22.2.8 (deprecated) evt_mesh_prov_heartbeat_publication_status

**Deprecated** and replaced by mesh_config_client_heartbeat_pub_status event.

Node heartbeat status, generated in response to a get heartbeat publication state or set heartbeat publication state request.

**Table 2.905. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The Network key index used in encrypting the response |
| 8-9 | uint16 | publication_address | Heartbeat publication address |
| 10 | uint8 | count_log | Heartbeat publication count setting. See set heartbeat publication state request for details. |
| 11 | uint8 | period_log | Heartbeat publication period setting. See set heartbeat publication state request for details. |
| 12 | uint8 | ttl | Time-to-live parameter for heartbeat messages |
| 13-14 | uint16 | features | Heartbeat trigger setting. See set heartbeat publication state request for details. |
| 15-16 | uint16 | publication_netkey_index | Index of the network key used to encrypt heartbeat messages |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_heartbeat_publication_status_id

/* Event structure */
struct gecko_msg_mesh_prov_heartbeat_publication_status_evt_t
{
  uint16 address;,
  uint16 netkey_index;,
  uint16 publication_address;,
  uint8 count_log;,
  uint8 period_log;,
  uint8 ttl;,
  uint16 features;,
  uint16 publication_netkey_index;
};
```

#### 2.22.2.9 (deprecated) evt_mesh_prov_heartbeat_subscription_status

**Deprecated** and replaced by mesh_config_client_heartbeat_sub_status event.

Node heartbeat subscription report.

**Table 2.906. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |
| 8-9 | uint16 | subscription_source | Source address for heartbeat messages |
| 10-11 | uint16 | subscription_destination | Destination address for heartbeat messages |
| 12 | uint8 | period_log | Heartbeat subscription remaining period. See heartbeat subscription set command for details. |
| 13 | uint8 | count_log | Binary logarithm of received heartbeat message count |
| 14 | uint8 | min_hops | Minimum hop value seen in received heartbeat messages |
| 15 | uint8 | max_hops | Maximum hop value seen in received heartbeat messages |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_heartbeat_subscription_status_id

/* Event structure */
struct gecko_msg_mesh_prov_heartbeat_subscription_status_evt_t
{
  uint16 address;,
  uint16 netkey_index;,
  uint16 subscription_source;,
  uint16 subscription_destination;,
  uint8 period_log;,
  uint8 count_log;,
  uint8 min_hops;,
  uint8 max_hops;
};
```

#### 2.22.2.10 evt_mesh_prov_initialized

Provisioner is initialized and operational.

**Table 2.907. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | networks | Number of network keys that the Provisioner has |
| 5-6 | uint16 | address | Unicast address of the primary element of the Provisioner |
| 7-10 | uint32 | ivi | IVI for network primary network (index 0) |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_initialized_id

/* Event structure */
struct gecko_msg_mesh_prov_initialized_evt_t
{
  uint8 networks;,
  uint16 address;,
  uint32 ivi;
};
```

#### 2.22.2.11 evt_mesh_prov_key_refresh_complete

Key refresh for a network key has completed

**Table 2.908. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | key | Network key index |
| 6-7 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_key_refresh_complete_id

/* Event structure */
struct gecko_msg_mesh_prov_key_refresh_complete_evt_t
{
  uint16 key;,
  uint16 result;
};
```

#### 2.22.2.12 evt_mesh_prov_key_refresh_node_update

Key refresh phase change for a node has occurred. This event is generated when a particular node has moved to a new key refresh phase.

**Table 2.909. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | key | Network key index |
| 6 | uint8 | phase | Phase moved into |
| 7 | uint8array | uuid | 16-byte UUID of the nod. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_key_refresh_node_update_id

/* Event structure */
struct gecko_msg_mesh_prov_key_refresh_node_update_evt_t
{
  uint16 key;,
  uint8 phase;,
  uint8array uuid;
};
```

#### 2.22.2.13 evt_mesh_prov_key_refresh_phase_update

Key refresh phase change for a network key has occurred. This event is generated when all nodes participating in a key refresh procedure have been moved to a new state (or have timed out, dropping them from the key refresh procedure).

**Table 2.910. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | key | Network key index |
| 6 | uint8 | phase | Phase moved into |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_key_refresh_phase_update_id

/* Event structure */
struct gecko_msg_mesh_prov_key_refresh_phase_update_evt_t
{
  uint16 key;,
  uint8 phase;
};
```

#### 2.22.2.14 (deprecated) evt_mesh_prov_model_pub_status

**Deprecated** and replaced by mesh_config_client_model_pub_status event.

**Table 2.911. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the queried model |
| 8-9 | uint16 | vendor_id | Vendor ID of the queried model. Returns 0xffff for Bluetooth SIG models. |
| 10-11 | uint16 | model_id | Model ID of the queried model |
| 12-13 | uint16 | appkey_index | The application key index to use for the published messages |
| 14-15 | uint16 | pub_address | The address to publish to |
| 16 | uint8 | ttl | Publication time-to-live value |
| 17 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 18 | uint8 | retrans | See documentation of model publication set command for details. |
| 19 | uint8 | credentials | Friendship credential flag. If the value is zero, publication is done using normal credentials. If the value is one, it is done with friendship credentials, meaning only the friend can decrypt the published message and relay it forward using the normal credentials. The default value is 0. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_model_pub_status_id

/* Event structure */
struct gecko_msg_mesh_prov_model_pub_status_evt_t
{
  uint16 result;,
  uint16 elem_address;,
  uint16 vendor_id;,
  uint16 model_id;,
  uint16 appkey_index;,
  uint16 pub_address;,
  uint8 ttl;,
```

```
  uint8 period;,
  uint8 retrans;,
  uint8 credentials;
};
```

#### 2.22.2.15 (deprecated) evt_mesh_prov_model_sub_addr

**Deprecated** and replaced by mesh_config_client_subs_list event.

This event is generated once for each subscription address a model reports when its subscription list is queried using the get subscription list command. The list is terminated with the subscription list entries end event.

**Table 2.912. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | elem_address | Unicast address of the element containing the queried model |
| 6-7 | uint16 | vendor_id | Vendor ID of the queried model. Returns 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID of the queried model. |
| 10-11 | uint16 | sub_addr | An address in the model subscription list |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_model_sub_addr_id

/* Event structure */
struct gecko_msg_mesh_prov_model_sub_addr_evt_t
{
  uint16 elem_address;,
  uint16 vendor_id;,
  uint16 model_id;,
  uint16 sub_addr;
};
```

#### 2.22.2.16 (deprecated) evt_mesh_prov_model_sub_addr_end

**Deprecated** and replace by mesh_config_client_subs_list_end event.

This event terminates model subscription list result reporting.

**Table 2.913. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | elem_address | Unicast address of the element containing the queried model |
| 8-9 | uint16 | vendor_id | Vendor ID of the queried model. Returns 0xffff for Bluetooth SIG models. |
| 10-11 | uint16 | model_id | Model ID of the queried model |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_model_sub_addr_end_id

/* Event structure */
struct gecko_msg_mesh_prov_model_sub_addr_end_evt_t
{
  uint16 result;,
  uint16 elem_address;,
  uint16 vendor_id;,
  uint16 model_id;
};
```

#### 2.22.2.17 (deprecated) evt_mesh_prov_network_list

**Deprecated** and replaced by mesh_config_client_netkey_list event.

Network key list event.

**Table 2.914. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the node |
| 6-7 | uint16 | netkey_index | Index of the network key |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_network_list_id

/* Event structure */
struct gecko_msg_mesh_prov_network_list_evt_t
{
  uint16 address;,
  uint16 netkey_index;
};
```

#### 2.22.2.18 (deprecated) evt_mesh_prov_network_list_end

**Deprecated** and replaced by mesh_config_client_netkey_list_end event.

Network key list terminator event.

**Table 2.915. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | address | Unicast address of the node |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_network_list_end_id

/* Event structure */
struct gecko_msg_mesh_prov_network_list_end_evt_t
{
  uint16 result;,
  uint16 address;
};
```

#### 2.22.2.19 (deprecated) evt_mesh_prov_node_reset

**Deprecated** and replaced by mesh_config_client_reset_status event.

A node has reset itself.

**Table 2.916. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the node |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_node_reset_id

/* Event structure */
struct gecko_msg_mesh_prov_node_reset_evt_t
{
  uint16 address;
};
```

#### 2.22.2.20 evt_mesh_prov_oob_auth_request

The Provisioner needs the device's output or static data. Provide it using prov_oob_auth_rsp.

**Table 2.917. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | output | Zero for static data, non-zero for output |
| 5 | uint8 | output_action | Output action type. Ignored for Static. |
| 6 | uint8 | output_size | Size of output data. Ignored for Static. |
| 7 | uint8array | uuid | UUID of the device |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_oob_auth_request_id

/* Event structure */
struct gecko_msg_mesh_prov_oob_auth_request_evt_t
{
  uint8 output;,
  uint8 output_action;,
  uint8 output_size;,
  uint8array uuid;
};
```

#### 2.22.2.21 evt_mesh_prov_oob_display_input

Random OOB input data was generated and should be displayed to and input with the device.

**Table 2.918. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | input_action | Input action type |
| 5 | uint8 | input_size | Number of digits or characters |
| 6 | uint8array | data | Raw 16-byte array |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_oob_display_input_id

/* Event structure */
struct gecko_msg_mesh_prov_oob_display_input_evt_t
{
  uint8 input_action;,
  uint8 input_size;,
  uint8array data;
};
```

#### 2.22.2.22 evt_mesh_prov_oob_pkey_request

The Provisioner needs the OOB public key of the Device with given UUID. Input the key using prov_oob_pkey_rsp.

**Table 2.919. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x06 | method | Message ID |
| 4 | uint8array | uuid | UUID of the Device |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_oob_pkey_request_id

/* Event structure */
struct gecko_msg_mesh_prov_oob_pkey_request_evt_t
{
  uint8array uuid;
};
```

#### 2.22.2.23 evt_mesh_prov_provisioning_failed

Provisioning a device failed.

**Table 2.920. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | reason | Reason for failure. Values are as follows:<br>• **0:** Link closed<br>• **1:** Invalid PDU<br>• **2:** Invalid PDU format<br>• **3:** Unexpected PDU<br>• **4:** Confirmation failed<br>• **5:** Out of resources<br>• **6:** Decryption failed<br>• **7:** Unexpected error<br>• **8:** Unable to assign address |
| 5 | uint8array | uuid | UUID of the device |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_provisioning_failed_id

/* Event structure */
struct gecko_msg_mesh_prov_provisioning_failed_evt_t
{
  uint8 reason;,
  uint8array uuid;
};
```

#### 2.22.2.24 (deprecated) evt_mesh_prov_relay_status

**Deprecated** and replaced by mesh_config_client_relay_status event.

Node relay state report.

**Table 2.921. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | address | Unicast address of the target node |
| 6-7 | uint16 | netkey_index | The network key index used in encrypting the request |
| 8 | uint8 | value | Relay state. Valid values are as follows:<br>• **0x00:** Relaying disabled<br>• **0x01:** Relaying enabled<br>• **0x02:** Relaying not supported |
| 9 | uint8 | count | Relay retransmit count. Value must be between 0 and 7. |
| 10 | uint8 | interval | Relay retransmit interval in milliseconds. Value is between 0 and 31; it represents 10-millisecond increments, starting at 10 ms. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_relay_status_id

/* Event structure */
struct gecko_msg_mesh_prov_relay_status_evt_t
{
  uint16 address;,
  uint16 netkey_index;,
  uint8 value;,
  uint8 count;,
  uint8 interval;
};
```

#### 2.22.2.25 evt_mesh_prov_unprov_beacon

Unprovisioned beacon seen.

**Table 2.922. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | | OOB capabilities bitfield, which indicates the means by which out-of-band provisioning data may be retrieved. |
| 6-9 | uint32 | uri_hash | Hash of the out-of-band URI, which is received in a separate event. If the URI bit (bit 1) is not set in the OOB capabilities bit-field, this field is ignored. |
| 10 | uint8 | bearer | Bearer on which the beacon was seen. Values are as follows:<br>• **0:** PB-ADV<br>• **1:** PB-GATT |
| 11-16 | bd_addr | address | Address of the device beaconing |
| 17 | uint8 | address_type | Address type of the device beaconing |
| 18 | uint8array | uuid | 16-byte UUID of the beaconing device. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_unprov_beacon_id

/* Event structure */
struct gecko_msg_mesh_prov_unprov_beacon_evt_t
{
  ;,
  uint32 uri_hash;,
  uint8 bearer;,
  bd_addr address;,
  uint8 address_type;,
  uint8array uuid;
};
```

#### 2.22.2.26 evt_mesh_prov_uri

URI advertisement received from a nearby device.

**Table 2.923. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x15 | class | Message class: Bluetooth Mesh Stack Provisioner |
| 3 | 0x0d | method | Message ID |
| 4-7 | uint32 | hash | URI hash. If a Provisioner is provisioning a device which supports out-of-band provisioning and has supplied a URI hash value in its Unprovisioned Device beacon, the Provisioner should check whether the hash matches this value. |
| 8 | uint8array | data | Raw URI data, formatted as specified in Bluetooth Core System Supplement v6. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_prov_uri_id

/* Event structure */
struct gecko_msg_mesh_prov_uri_evt_t
{
  uint32 hash;,
  uint8array data;
};
```

### 2.22.3 mesh_prov defines

#### 2.22.3.1 define_mesh_prov_oob_capabilities

OOB capability bitmask constants

**Table 2.924. Defines**

| Value | Name | Description |
|---|---|---|
| 1 | MESH_PROV_OOB_OTHER | Uncategorized |
| 2 | MESH_PROV_OOB_URI | URI or other electronic |
| 4 | MESH_PROV_OOB_2D_MR_CODE | 2D machine-readable code |
| 8 | MESH_PROV_OOB_BAR_CODE | Barcode |
| 16 | MESH_PROV_OOB_NFC | NFC |
| 32 | MESH_PROV_OOB_NUMBER | Number |
| 64 | MESH_PROV_OOB_STRING | String |
| 128 | MESH_PROV_OOB_RFU_7 | Reserved |
| 256 | MESH_PROV_OOB_RFU_8 | Reserved |
| 512 | MESH_PROV_OOB_RFU_9 | Reserved |
| 1024 | MESH_PROV_OOB_RFU_A | Reserved |
| 2048 | MESH_PROV_OOB_LOC_ON_BOX | On the box |
| 4096 | MESH_PROV_OOB_LOC_IN_BOX | Inside the box |
| 8192 | MESH_PROV_OOB_LOC_PAPER | On a piece of paper |
| 16384 | MESH_PROV_OOB_LOC_MANUAL | In the device manual |
| 32768 | MESH_PROV_OOB_LOC_DEVICE | On the device |
| 1920 | MESH_PROV_OOB_RFU_MASK | Mask of reserved bits |

## 2.23 Bluetooth Mesh Proxy Connections (mesh_proxy)

Bluetooth mesh stack functions for GATT proxy connections

### 2.23.1 mesh_proxy commands

#### 2.23.1.1 cmd_mesh_proxy_allow

Allow messages destined to the given address to be forwarded over the proxy connection to the proxy client. At the proxy server side, this is a local configuration, while on the proxy client a proxy configuration PDU will be sent to the proxy server.

**Table 2.925. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x03 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |
| 8-9 | uint16 | address | Destination address to allow. The address may be either a unicast address, a group address, or a virtual address. |
| 10-11 | uint16 | key | Network key index used in encrypting the request to the proxy server |

**Table 2.926. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_proxy_allow_rsp_t *gecko_cmd_mesh_proxy_allow(uint32 handle, uint16 address, uint16 key);

/* Response id */
gecko_rsp_mesh_proxy_allow_id

/* Response structure */
struct gecko_msg_mesh_proxy_allow_rsp_t
{
  uint16 result;
};
```

#### 2.23.1.2 cmd_mesh_proxy_connect

Start connecting a proxy client to a proxy server. After the connection is complete, a connection established event will be generated. LE-connection must be opened prior to opening proxy connection.

**Table 2.927. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.928. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | handle | If a connection attempt is successfully initiated, a valid proxy handle will be returned. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_proxy_connect_rsp_t *gecko_cmd_mesh_proxy_connect(uint8 connection);

/* Response id */
gecko_rsp_mesh_proxy_connect_id

/* Response structure */
struct gecko_msg_mesh_proxy_connect_rsp_t
{
  uint16 result;,
  uint32 handle;
};
```

#### 2.23.1.3 cmd_mesh_proxy_deny

Block messages meant for the given address from being forwarded over the proxy connection to the proxy client. At the proxy server side, this is a local configuration, while on the proxy client a proxy configuration PDU will be sent to the proxy server.

**Table 2.929. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x04 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |
| 8-9 | uint16 | address | Destination address to block. The address may be either a unicast address, a group address, or a virtual address. |
| 10-11 | uint16 | key | Network key index used in encrypting the request to the proxy server |

**Table 2.930. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_proxy_deny_rsp_t *gecko_cmd_mesh_proxy_deny(uint32 handle, uint16 address, uint16 key);

/* Response id */
gecko_rsp_mesh_proxy_deny_id

/* Response structure */
struct gecko_msg_mesh_proxy_deny_rsp_t
{
  uint16 result;
};
```

#### 2.23.1.4 cmd_mesh_proxy_disconnect

Disconnect. This call can be used also for a connection, which is not yet fully formed.

**Table 2.931. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x01 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |

**Table 2.932. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_proxy_disconnect_rsp_t *gecko_cmd_mesh_proxy_disconnect(uint32 handle);

/* Response id */
gecko_rsp_mesh_proxy_disconnect_id

/* Response structure */
struct gecko_msg_mesh_proxy_disconnect_rsp_t
{
  uint16 result;
};
```

#### 2.23.1.5 cmd_mesh_proxy_optimisation_toggle

In case of unicast address, if proxy identified the destination, the message will be forwarded only to that node, otherwise to all. This functionality could be enabled or disabled with this function.

**Table 2.933. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | enable | Non zero - enable, otherwise disable |

**Table 2.934. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_proxy_optimisation_toggle_rsp_t *gecko_cmd_mesh_proxy_optimisation_toggle(uint8 enable);

/* Response id */
gecko_rsp_mesh_proxy_optimisation_toggle_id

/* Response structure */
struct gecko_msg_mesh_proxy_optimisation_toggle_rsp_t
{
  uint16 result;
};
```

#### 2.23.1.6 cmd_mesh_proxy_set_filter_type

Set up proxy filtering type. At the proxy server side, this is a local configuration, while on the proxy client a proxy configuration PDU will be sent to the proxy server.

**Table 2.935. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x02 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |
| 8 | uint8 | type | Filter type: 0x00 for whitelist, 0x01 for blacklist |
| 9-10 | uint16 | key | Network key index used in encrypting the request to the proxy server |

**Table 2.936. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_proxy_set_filter_type_rsp_t *gecko_cmd_mesh_proxy_set_filter_type(uint32 handle, uint8
type, uint16 key);

/* Response id */
gecko_rsp_mesh_proxy_set_filter_type_id

/* Response structure */
struct gecko_msg_mesh_proxy_set_filter_type_rsp_t
{
  uint16 result;
};
```

### 2.23.2 mesh_proxy events

#### 2.23.2.1 evt_mesh_proxy_connected

Indication that a connection has been successfully formed.

**Table 2.937. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |

**C Functions**

```
/* Event id */
gecko_evt_mesh_proxy_connected_id

/* Event structure */
struct gecko_msg_mesh_proxy_connected_evt_t
{
  uint32 handle;
};
```

#### 2.23.2.2 evt_mesh_proxy_disconnected

Indication that a connection has been disconnected or a connection attempt failed.

**Table 2.938. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x01 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |
| 8-9 | uint16 | reason | Reason for disconnection |

**C Functions**

```
/* Event id */
gecko_evt_mesh_proxy_disconnected_id

/* Event structure */
struct gecko_msg_mesh_proxy_disconnected_evt_t
{
  uint32 handle;,
  uint16 reason;
};
```

#### 2.23.2.3 evt_mesh_proxy_filter_status

Proxy status report event

**Table 2.939. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x18 | class | Message class: Bluetooth Mesh Proxy Connections |
| 3 | 0x02 | method | Message ID |
| 4-7 | uint32 | handle | Proxy handle |
| 8 | uint8 | type | Filter type: 0x00 for whitelist, 0x01 for blacklist. |
| 9-10 | uint16 | count | Current filter list length |

**C Functions**

```
/* Event id */
gecko_evt_mesh_proxy_filter_status_id

/* Event structure */
struct gecko_msg_mesh_proxy_filter_status_evt_t
{
  uint32 handle;,
  uint8 type;,
  uint16 count;
};
```

### 2.24 Bluetooth Mesh GATT Proxy Client (mesh_proxy_client)

Initialize the GATT Proxy client-side functionality. Mesh proxy commands are in the mesh_proxy class. This class allows the linker to drop the GATT Proxy client code if it is not needed. It is enough to initialize this BGAPI class. It contains no commands or events.

### 2.25 Bluetooth Mesh GATT Proxy Server (mesh_proxy_server)

Initialize the GATT Proxy server-side functionality. This class allows the linker to drop the GATT Proxy server code if it is not needed. It is enough to initialize this BGAPI class. It contains no commands or events.

## 2.26 Bluetooth Mesh Scene Client Model (mesh_scene_client)

Bluetooth Mesh Scene Client model functionality to send and receive messages to/from the Scene Server and Scene Setup Server models.

Throughout the API, the client model being used is identified by its element address and model ID, while the server model responding to client model requests is identified by its element address and model ID.

The API has functions for querying server model states and requesting server model state changes

### 2.26.1 mesh_scene_client commands

#### 2.26.1.1 cmd_mesh_scene_client_delete

Delete a scene.

**Table 2.940. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11-12 | uint16 | selected_scene | Scene of interest |

**Table 2.941. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scene_client_delete_rsp_t *gecko_cmd_mesh_scene_client_delete(uint16 elem_index, uint16
server_address, uint16 appkey_index, uint8 flags, uint16 selected_scene);

/* Response id */
gecko_rsp_mesh_scene_client_delete_id

/* Response structure */
struct gecko_msg_mesh_scene_client_delete_rsp_t
{
  uint16 result;
};
```

#### 2.26.1.2 cmd_mesh_scene_client_get

Scene Get command.

**Table 2.942. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |
| 6-7 | uint16 | server_address | Device to be queried. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | Appkey used by server_address. |

**Table 2.943. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scene_client_get_rsp_t  *gecko_cmd_mesh_scene_client_get(uint16  elem_index,  uint16
server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_scene_client_get_id

/* Response structure */
struct gecko_msg_mesh_scene_client_get_rsp_t
{
  uint16 result;
};
```

**Table 2.944. Events Generated**

| Event | Description |
|---|---|
| mesh_scene_client_status | Event indicating an incoming Scene Status message. |

#### 2.26.1.3 cmd_mesh_scene_client_get_register

Scene Register Get command

**Table 2.945. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |

**Table 2.946. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_scene_client_get_register_rsp_t    *gecko_cmd_mesh_scene_client_get_register(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_scene_client_get_register_id

/* Response structure */
struct gecko_msg_mesh_scene_client_get_register_rsp_t
{
  uint16 result;
};
```

**Table 2.947. Events Generated**

| Event | Description |
|---|---|
| mesh_scene_client_register_status | Event indicating an incoming Scene Register Status message. |

#### 2.26.1.4 cmd_mesh_scene_client_init

Initializes the Scene Client model. Scene Client does not have any internal configuration, it only activates the model in the mesh stack.

**Table 2.948. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |

**Table 2.949. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scene_client_init_rsp_t *gecko_cmd_mesh_scene_client_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scene_client_init_id

/* Response structure */
struct gecko_msg_mesh_scene_client_init_rsp_t
{
  uint16 result;
};
```

#### 2.26.1.5 cmd_mesh_scene_client_recall

Recall a scene.

**Table 2.950. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the client element. |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | Appkey used by server_address. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11-12 | uint16 | selected_scene | Scene of interest |
| 13 | uint8 | tid | Transaction ID |
| 14-17 | uint32 | transition_time | Amount of time (in milliseconds) allotted for the transition to take place. Value of 0xFFFFFFFF will cause this parameter as well as the "delay" parameter to be omitted. The transition will be immediate if both the transition time and the delay are zero. |
| 18-21 | uint32 | delay | Message execution delay in milliseconds. If the "transition_time" is 0xFFFFFFFF, this parameter is ignored. If both the transition time and the delay are zero the transition is immediate. |

**Table 2.951. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scene_client_recall_rsp_t *gecko_cmd_mesh_scene_client_recall(uint16 elem_index, uint16
server_address, uint16 appkey_index, uint8 flags, uint16 selected_scene, uint8 tid, uint32 transition_time,
uint32 delay);

/* Response id */
gecko_rsp_mesh_scene_client_recall_id
```

```
/* Response structure */
struct gecko_msg_mesh_scene_client_recall_rsp_t
{
  uint16 result;
};
```

#### 2.26.1.6 cmd_mesh_scene_client_store

Store a scene.

**Table 2.952. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET PROPERTY message will be sent, zero will send SET PROPERTY UNACKNOWLEDGED |
| 11-12 | uint16 | selected_scene | Scene of interest |

**Table 2.953. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scene_client_store_rsp_t  *gecko_cmd_mesh_scene_client_store(uint16  elem_index,  uint16
server_address, uint16 appkey_index, uint8 flags, uint16 selected_scene);

/* Response id */
gecko_rsp_mesh_scene_client_store_id

/* Response structure */
struct gecko_msg_mesh_scene_client_store_rsp_t
{
  uint16 result;
};
```

### 2.26.2 mesh_scene_client events

#### 2.26.2.1 evt_mesh_scene_client_register_status

Event indicating an incoming Scene Register Status message.

**Table 2.954. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element for which received the status. |
| 6-7 | uint16 | server_address | Device which sent the status. |
| 8-9 | uint16 | destination_address | Address of the client or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |
| 12 | uint8 | status | Status of the request. |
| 13-14 | uint16 | current_scene | Currently selected scene. |
| 15-16 | uint16array | scenes | List of all the scenes on the server. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_client_register_status_id

/* Event structure */
struct gecko_msg_mesh_scene_client_register_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 status;,
  uint16 current_scene;,
  uint16array scenes;
};
```

#### 2.26.2.2 evt_mesh_scene_client_status

Event indicating an incoming Scene Status message.

**Table 2.955. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x4f | class | Message class: Bluetooth Mesh Scene Client Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the status. |
| 6-7 | uint16 | server_address | Device which sent the status. |
| 8-9 | uint16 | destination_address | Address of the client or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |
| 12 | uint8 | status | Status of the request. |
| 13-14 | uint16 | current_scene | Currently selected scene. |
| 15-16 | uint16 | target_scene | Scene to be transitioning to. |
| 17-20 | uint32 | remaining_time | Time (in milliseconds) remaining in transition. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_client_status_id

/* Event structure */
struct gecko_msg_mesh_scene_client_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint8 status;,
  uint16 current_scene;,
  uint16 target_scene;,
  uint32 remaining_time;
};
```

## 2.27 Bluetooth Mesh Scene Server Model (mesh_scene_server)

Bluetooth Mesh Scene model Server functionality.

### 2.27.1 mesh_scene_server commands

#### 2.27.1.1 cmd_mesh_scene_server_deinit

De-initializes the Scene Server model.

**Table 2.956. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.957. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scene_server_deinit_rsp_t *gecko_cmd_mesh_scene_server_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scene_server_deinit_id

/* Response structure */
struct gecko_msg_mesh_scene_server_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.27.1.2 cmd_mesh_scene_server_enable_compact_recall_events

Switch to compact reporting for recalled states. Compact state reduces amount buffering memory needed by the scene recall and is recommended for devices with big amount of models or for devices in environment with lots of bluetooth advertisement traffic.

When compact mode is active @ref sl_btmesh_evt_scene_server_compact_recall is generated instead of several @ref sl_btmesh_evt_generic_server_state_recall events.

**Table 2.958. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x03 | method | Message ID |

**Table 2.959. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_scene_server_enable_compact_recall_events_rsp_t
*gecko_cmd_mesh_scene_server_enable_compact_recall_events();

/* Response id */
gecko_rsp_mesh_scene_server_enable_compact_recall_events_id

/* Response structure */
struct gecko_msg_mesh_scene_server_enable_compact_recall_events_rsp_t
{
  uint16 result;
};
```

#### 2.27.1.3 cmd_mesh_scene_server_init

Initializes the Scene Server model. Server does not have any internal configuration, command only activates the model in the mesh stack.

**Table 2.960. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.961. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scene_server_init_rsp_t *gecko_cmd_mesh_scene_server_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scene_server_init_id

/* Response structure */
struct gecko_msg_mesh_scene_server_init_rsp_t
{
  uint16 result;
};
```

#### 2.27.1.4 cmd_mesh_scene_server_reset_register

Reset register value. This commmand should be invoked if state of a model has been modified in such a manner that it cannot no longer considered to be in scene indicated by the scene register.

**Table 2.962. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. This can be either element of the updated model or the element of the scene model responsible for controlling scene of the updated model. |

**Table 2.963. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scene_server_reset_register_rsp_t  *gecko_cmd_mesh_scene_server_reset_register(uint16
elem_index);

/* Response id */
gecko_rsp_mesh_scene_server_reset_register_id

/* Response structure */
struct gecko_msg_mesh_scene_server_reset_register_rsp_t
{
  uint16 result;
};
```

### 2.27.2 mesh_scene_server events

#### 2.27.2.1 evt_mesh_scene_server_compact_recall

Recall a scene.

**Table 2.964. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x05 | method | Message ID |
| 4 | uint8array | states | Byte array containind recalled states. Array consist of 5 byte chunks as follows:<br>• Element id as 16bit unsigned integer in little endian format<br>• Model id as 16bit unsigned integer in little endian format<br>• Model-specific state type, identifying the kind of state recalled See get state types list for details.<br>after this event application can request recalled states with @ref sl_btmesh_generic_server_get_cached_state command |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_server_compact_recall_id

/* Event structure */
struct gecko_msg_mesh_scene_server_compact_recall_evt_t
{
  uint8array states;
};
```

#### 2.27.2.2 evt_mesh_scene_server_get

Get the status.

**Table 2.965. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the command. |
| 6-7 | uint16 | client_address | Device which sent the request. |
| 8-9 | uint16 | destination_address | Address of the server or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_server_get_id

/* Event structure */
struct gecko_msg_mesh_scene_server_get_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;
};
```

#### 2.27.2.3 evt_mesh_scene_server_publish

Indicates that the publishing period timer elapsed and the app should/can publish its state or any request.

**Table 2.966. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-9 | uint32 | period_ms | The current publishing period that can be used to estimate the next tick, e.g., when the state should be reported at higher frequency. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_server_publish_id

/* Event structure */
struct gecko_msg_mesh_scene_server_publish_evt_t
{
  uint16 elem_index;,
  uint32 period_ms;
};
```

#### 2.27.2.4 evt_mesh_scene_server_recall

Recall a scene.

**Table 2.967. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the command. |
| 6-7 | uint16 | client_address | Device which sent the request. |
| 8-9 | uint16 | destination_address | Address of the server or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |
| 12-13 | uint16 | selected_scene | Scene of interest |
| 14-17 | uint32 | transition_time | Time (in milliseconds) allotted for the transition to take place |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_server_recall_id

/* Event structure */
struct gecko_msg_mesh_scene_server_recall_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint16 selected_scene;,
  uint32 transition_time;
};
```

#### 2.27.2.5 evt_mesh_scene_server_register_get

Get the status of a register

**Table 2.968. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x50 | class | Message class: Bluetooth Mesh Scene Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the command. |
| 6-7 | uint16 | client_address | Device which sent the request. |
| 8-9 | uint16 | destination_address | Address of the server or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_server_register_get_id

/* Event structure */
struct gecko_msg_mesh_scene_server_register_get_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;
};
```

## 2.28 Bluetooth Mesh Scene Setup Server Model (mesh_scene_setup_server)

Bluetooth Mesh Scene model Setup Server functionality.

### 2.28.1 mesh_scene_setup_server commands

#### 2.28.1.1 cmd_mesh_scene_setup_server_init

Initializes the Scene Setup Server model. Server does not have any internal configuration, command only activates the model in the mesh stack.

**Table 2.969. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x51 | class | Message class: Bluetooth Mesh Scene Setup Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element. |

**Table 2.970. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x51 | class | Message class: Bluetooth Mesh Scene Setup Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scene_setup_server_init_rsp_t *gecko_cmd_mesh_scene_setup_server_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scene_setup_server_init_id

/* Response structure */
struct gecko_msg_mesh_scene_setup_server_init_rsp_t
{
  uint16 result;
};
```

### 2.28.2 mesh_scene_setup_server events

#### 2.28.2.1 evt_mesh_scene_setup_server_delete

Scene Delete from the Client

**Table 2.971. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x51 | class | Message class: Bluetooth Mesh Scene Setup Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the command. |
| 6-7 | uint16 | client_address | Device which sent the request. |
| 8-9 | uint16 | destination_address | Address of the server or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |
| 12-13 | uint16 | scene_id | Scene ID |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_setup_server_delete_id

/* Event structure */
struct gecko_msg_mesh_scene_setup_server_delete_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint16 scene_id;
};
```

#### 2.28.2.2 evt_mesh_scene_setup_server_publish

Indicates that the publishing period timer elapsed and the app should/can publish its state or any request.

**Table 2.972. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x51 | class | Message class: Bluetooth Mesh Scene Setup Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-9 | uint32 | period_ms | The current publishing period that can be used to estimate the next tick, e.g., when the state should be reported at higher frequency. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_setup_server_publish_id

/* Event structure */
struct gecko_msg_mesh_scene_setup_server_publish_evt_t
{
  uint16 elem_index;,
  uint32 period_ms;
};
```

#### 2.28.2.3 evt_mesh_scene_setup_server_store

Scene Store from the Client

**Table 2.973. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x51 | class | Message class: Bluetooth Mesh Scene Setup Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Index of the element which received the command. |
| 6-7 | uint16 | client_address | Device which sent the request. |
| 8-9 | uint16 | destination_address | Address of the server or the group address to which it was published. |
| 10-11 | uint16 | appkey_index | Appkey used by client_address. |
| 12-13 | uint16 | scene_id | Scene ID |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scene_setup_server_store_id

/* Event structure */
struct gecko_msg_mesh_scene_setup_server_store_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 destination_address;,
  uint16 appkey_index;,
  uint16 scene_id;
};
```

## 2.29 Bluetooth Mesh Scheduler Client (mesh_scheduler_client)

This class provides commands and messages to interface with the Scheduler Client model. Scheduler models uses multiple fields to define the occurence of an event and the type of event to be triggered.

For the description of these fields, please see

### 2.29.1 mesh_scheduler_client commands

#### 2.29.1.1 cmd_mesh_scheduler_client_deinit

Deinitializes the Scheduler Client model

**Table 2.974. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |

**Table 2.975. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scheduler_client_deinit_rsp_t *gecko_cmd_mesh_scheduler_client_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scheduler_client_deinit_id

/* Response structure */
struct gecko_msg_mesh_scheduler_client_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.29.1.2 cmd_mesh_scheduler_client_get

Send a Scheduler Get message to get the Scheduler Register status of a Scheduler Server

**Table 2.976. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |

**Table 2.977. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scheduler_client_get_rsp_t  *gecko_cmd_mesh_scheduler_client_get(uint16  elem_index,
uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_scheduler_client_get_id

/* Response structure */
struct gecko_msg_mesh_scheduler_client_get_rsp_t
{
  uint16 result;
};
```

**Table 2.978. Events Generated**

| Event | Description |
|---|---|
| mesh_scheduler_client_status | Scheduler Status response message to a Scheduler Get/Set command |

#### 2.29.1.3 cmd_mesh_scheduler_client_get_action

Send a Scheduler Action Get message to get action defined by a Schedule Register state entry

**Table 2.979. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | index | Index of the Scgeduler Register entry to get |

**Table 2.980. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scheduler_client_get_action_rsp_t  *gecko_cmd_mesh_scheduler_client_get_action(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 index);

/* Response id */
gecko_rsp_mesh_scheduler_client_get_action_id

/* Response structure */
struct gecko_msg_mesh_scheduler_client_get_action_rsp_t
{
  uint16 result;
};
```

**Table 2.981. Events Generated**

| Event | Description |
|---|---|
| mesh_scheduler_client_action_status | Scheduler Status response message to a Scheduler Get/Set command<br>For the description of these fields, please see |

#### 2.29.1.4 cmd_mesh_scheduler_client_init

Initializes the Scheduler Client model

**Table 2.982. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |

**Table 2.983. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scheduler_client_init_rsp_t *gecko_cmd_mesh_scheduler_client_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scheduler_client_init_id

/* Response structure */
struct gecko_msg_mesh_scheduler_client_init_rsp_t
{
  uint16 result;
};
```

#### 2.29.1.5 cmd_mesh_scheduler_client_set_action

Send a Scheduler Action Set message to set the given entry of the Scheduler Register state

For the description of these fields, please see

**Table 2.984. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x14 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required, othewise the unacknowledged message is used. |
| 11 | uint8 | index | Index of the Scheduler Register entry to set |
| 12 | uint8 | year | Scheduled year for the action |
| 13-14 | uint16 | month | Scheduled month for the action |
| 15 | uint8 | day | Scheduled day of the month for the action |
| 16 | uint8 | hour | Scheduled hour for the action |
| 17 | uint8 | minute | Scheduled minute for the action |
| 18 | uint8 | second | Scheduled second for the action |
| 19 | uint8 | day_of_week | Scheduled days of the week for the action |
| 20 | uint8 | action | Action to be performed at the scheduled time |
| 21 | uint8 | transition_time | Transition time for this action |
| 22-23 | uint16 | scene_number | Scene number to be used for some actions |

**Table 2.985. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scheduler_client_set_action_rsp_t  *gecko_cmd_mesh_scheduler_client_set_action(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint8 index, uint8 year, uint16 month,
uint8 day, uint8 hour, uint8 minute, uint8 second, uint8 day_of_week, uint8 action, uint8 transition_time,
uint16 scene_number);

/* Response id */
gecko_rsp_mesh_scheduler_client_set_action_id

/* Response structure */
struct gecko_msg_mesh_scheduler_client_set_action_rsp_t
{
  uint16 result;
};
```

**Table 2.986. Events Generated**

| Event | Description |
|---|---|
| mesh_scheduler_client_action_status | Scheduler Status response message to a Scheduler Get/Set command<br><br>For the description of these fields, please see |

### 2.29.2 mesh_scheduler_client events

#### 2.29.2.1 evt_mesh_scheduler_client_action_status

Scheduler Status response message to a Scheduler Get/Set command

For the description of these fields, please see

**Table 2.987. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x15 | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use. |
| 12 | uint8 | index | Index of the Scheduler Register |
| 13 | uint8 | year | Scheduled year for the action |
| 14-15 | uint16 | month | Scheduled month for the action |
| 16 | uint8 | day | Scheduled day of the month for the action |
| 17 | uint8 | hour | Scheduled hour for the action |
| 18 | uint8 | minute | Scheduled minute for the action |
| 19 | uint8 | second | Scheduled second for the action |
| 20 | uint8 | day_of_week | Scheduled days of the week for the action |
| 21 | uint8 | action | Action to be performed at the scheduled time |
| 22 | uint8 | transition_time | Transition time for this action |
| 23-24 | uint16 | scene_number | Scene number to be used for some actions |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scheduler_client_action_status_id

/* Event structure */
struct gecko_msg_mesh_scheduler_client_action_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 index;,
  uint8 year;,
  uint16 month;,
  uint8 day;,
  uint8 hour;,
  uint8 minute;,
  uint8 second;,
  uint8 day_of_week;,
  uint8 action;,
  uint8 transition_time;,
```

```
  uint16 scene_number;
};
```

#### 2.29.2.2 evt_mesh_scheduler_client_status

Scheduler Status response message to a Scheduler Get/Set command

**Table 2.988. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x54 | class | Message class: Bluetooth Mesh Scheduler Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use. |
| 12-13 | uint16 | scheduler | Bit field indicating defined Actions in the Schedule Register |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scheduler_client_status_id

/* Event structure */
struct gecko_msg_mesh_scheduler_client_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint16 scheduler;
};
```

## 2.30 Bluetooth Mesh Scheduler Server (mesh_scheduler_server)

This class provides commands and messages to interface with the Scheduler Server model

Scheduler server uses multiple fields to define the occurence of an event and the type of event to be triggered. The field definitions are as follows.

**Year**
The year field represents 2 last significant digits of the year of the occurence of the scheduled event.
- 0x00-0x63: 2 least significant digits of the year (0-99)
- 0x64: Any year
- All other values are prohibited

**Month**

The month field represents the months of the occurences of the scheduled event, using a bitfield with one bit for each month.
- Bit 0: Scheduled in January
- Bit 1: February
- Bit 2: March
- Bit 3: April
- Bit 4: May
- Bit 5: June
- Bit 6: July
- Bit 7: August
- Bit 8: September
- Bit 9: October
- Bit 10: November
- Bit 11: December

**Day**

The Day field represents the day of the month of the occurence of the scheduled event. If the day of the month has a number that is larger than the number of days in the month, then the event occurs in the last day of the month. For example, in February if the day field holds the value 29, the action is triggered on February 28th in a non-leap year or February 29th in a leap year.
- 0x00: Any day
- 0x01-0x1F: Day of the month (1-31)
- All other values are prohibited

**Hour**

The Hour field represents the hour of the occurence of the scheduled event.
- 0x00-0x17: Hour of the day (0-23)
- 0x18: Any hour of the day
- 0x19: Once a day (at a random hour)
- All other values are prohibited

**Minute**

The Minute field represents the minute of the occurence of the scheduled event.
- 0x00-0x3B: Minute of the hour (0-59)
- 0x3C: Any minute of the hour
- 0x3D: Every 15 minutes (0, 15, 30, 45)
- 0x3E: Every 20 minutes (0, 20, 40)
- 0x3F: Once an hour (at a random minute)
- All other values are prohibited

**Second**

The Second field represents the second of the occurence of the scheduled event.
- 0x00-0x3B: Seconds of the minute (0-59)
- 0x3C: Any second of the minute

- 0x3D: Every 15 seconds (0, 15, 30, 45)
- 0x3E: Every 20 seconds (0, 20, 40)
- 0x3F: Once a minute (at a random second)
- All other values are prohibited

**Day of Week**

The DayOfWeek field represents the days of the week when the scheduled event will trigger. The week days are represented by a bitfield, by one bit for each day.

- Bit 0: Scheduled on Mondays
- Bit 1: Scheduled on Tuesdays
- Bit 2: Scheduled on Wednesdays
- Bit 3: Scheduled on Thursdays
- Bit 4: Scheduled on Fridays
- Bit 5: Scheduled on Saturdays
- Bit 6: Scheduled on Sundays

**Action**

The action field represents an action to be executed for a scheduled event

- 0x00: Turn Off
- 0x01: Turn On
- 0x02: Scene Recall
- 0x0F: No action
- All other values are prohibited

**Transition time**

This is a 1-octet value that consists of two fields: a 2-bit bit field representing the step resolution and a 6-bit bit field representing the number of transition steps.

Bit 0-1: Transition Step Resolution

- 0b00: The Default Transition Step Resolution is 100 milliseconds
- 0b01: 1 second resolution
- 0b10: 10 seconds resolution
- 0b11: 10 minutes resolution

Bit 2-7: Transition Number of Steps

- 0x00: The Transition Time is immediate
- 0x01-0x3E: The number of steps
- 0x3F: The value is unknown. The state cannot be set to this value, but an element may report an unknown value if a transition is higher than 0x3E or not determined

### 2.30.1 mesh_scheduler_server commands

#### 2.30.1.1 cmd_mesh_scheduler_server_deinit

Deinitializes the Scheduler Server model

**Table 2.989. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |

**Table 2.990. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scheduler_server_deinit_rsp_t *gecko_cmd_mesh_scheduler_server_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scheduler_server_deinit_id

/* Response structure */
struct gecko_msg_mesh_scheduler_server_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.30.1.2 cmd_mesh_scheduler_server_get

Get Scheduler Register status of Scheduler Server

**Table 2.991. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |

**Table 2.992. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | status | Unsigned 16-bit integer |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scheduler_server_get_rsp_t *gecko_cmd_mesh_scheduler_server_get(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scheduler_server_get_id

/* Response structure */
struct gecko_msg_mesh_scheduler_server_get_rsp_t
{
  uint16 result;,
  uint16 status;
};
```

#### 2.30.1.3 cmd_mesh_scheduler_server_get_action

Get the Scheduler Action defined by a Schedule Register state entry.

For the description of returned fields, please see

**Table 2.993. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |
| 6 | uint8 | index | Index of the Scgeduler Register entry to get |

**Table 2.994. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | index | Index of the Scheduler Register entry to set |
| 7 | uint8 | year | Scheduled year for the action |
| 8-9 | uint16 | month | Scheduled month for the action |
| 10 | uint8 | day | Scheduled day of the month for the action |
| 11 | uint8 | hour | Scheduled hour for the action |
| 12 | uint8 | minute | Scheduled minute for the action |
| 13 | uint8 | second | Scheduled second for the action |
| 14 | uint8 | day_of_week | Scheduled days of the week for the action |
| 15 | uint8 | action | Action to be performed at the scheduled time |
| 16 | uint8 | transition_time | Transition time for this action |
| 17-18 | uint16 | scene_number | Scene number to be used for some actions |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scheduler_server_get_action_rsp_t  *gecko_cmd_mesh_scheduler_server_get_action(uint16
elem_index, uint8 index);

/* Response id */
gecko_rsp_mesh_scheduler_server_get_action_id
```

```
/* Response structure */
struct gecko_msg_mesh_scheduler_server_get_action_rsp_t
{
  uint16 result;,
  uint8 index;,
  uint8 year;,
  uint16 month;,
  uint8 day;,
  uint8 hour;,
  uint8 minute;,
  uint8 second;,
  uint8 day_of_week;,
  uint8 action;,
  uint8 transition_time;,
  uint16 scene_number;
};
```

#### 2.30.1.4 cmd_mesh_scheduler_server_init

Initializes the Scheduler Server model

**Table 2.995. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |

**Table 2.996. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_scheduler_server_init_rsp_t *gecko_cmd_mesh_scheduler_server_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_scheduler_server_init_id

/* Response structure */
struct gecko_msg_mesh_scheduler_server_init_rsp_t
{
  uint16 result;
};
```

#### 2.30.1.5 cmd_mesh_scheduler_server_set_action

Set the given Scheduler Action entry of the Scheduler Register state

For the description of these fields, please see

**Table 2.997. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |
| 6 | uint8 | index | Index of the Scheduler Register entry to set |
| 7 | uint8 | year | Scheduled year for the action |
| 8-9 | uint16 | month | Scheduled month for the action |
| 10 | uint8 | day | Scheduled day of the month for the action |
| 11 | uint8 | hour | Scheduled hour for the action |
| 12 | uint8 | minute | Scheduled minute for the action |
| 13 | uint8 | second | Scheduled second for the action |
| 14 | uint8 | day_of_week | Scheduled days of the week for the action |
| 15 | uint8 | action | Action to be performed at the scheduled time |
| 16 | uint8 | transition_time | Transition time for this action |
| 17-18 | uint16 | scene_number | Scene number to be used for some actions |

**Table 2.998. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_scheduler_server_set_action_rsp_t  *gecko_cmd_mesh_scheduler_server_set_action(uint16
elem_index, uint8 index, uint8 year, uint16 month, uint8 day, uint8 hour, uint8 minute, uint8 second, uint8
day_of_week, uint8 action, uint8 transition_time, uint16 scene_number);

/* Response id */
gecko_rsp_mesh_scheduler_server_set_action_id
```

```
/* Response structure */
struct gecko_msg_mesh_scheduler_server_set_action_rsp_t
{
  uint16 result;
};
```

### 2.30.2 mesh_scheduler_server events

#### 2.30.2.1 evt_mesh_scheduler_server_action_changed

Notification of a Scheduler Action change as the result of a Scheduler Set message

For the description of these fields, please see

**Table 2.999. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |
| 6 | uint8 | index | Index of the Scheduler Register |
| 7 | uint8 | year | Scheduled year for the action |
| 8-9 | uint16 | month | Scheduled month for the action |
| 10 | uint8 | day | Scheduled day of the month for the action |
| 11 | uint8 | hour | Scheduled hour for the action |
| 12 | uint8 | minute | Scheduled minute for the action |
| 13 | uint8 | second | Scheduled second for the action |
| 14 | uint8 | day_of_week | Scheduled days of the week for the action |
| 15 | uint8 | action | Action to be performed at the scheduled time |
| 16 | uint8 | transition_time | Transition time for this action |
| 17-18 | uint16 | scene_number | Scene number to be used for some actions |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scheduler_server_action_changed_id

/* Event structure */
struct gecko_msg_mesh_scheduler_server_action_changed_evt_t
{
  uint16 elem_index;,
  uint8 index;,
  uint8 year;,
  uint16 month;,
  uint8 day;,
  uint8 hour;,
  uint8 minute;,
  uint8 second;,
  uint8 day_of_week;,
  uint8 action;,
  uint8 transition_time;,
  uint16 scene_number;
};
```

#### 2.30.2.2 evt_mesh_scheduler_server_scene_changed

Notification that scheduled scene change took place

**Table 2.1000. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x55 | class | Message class: Bluetooth Mesh Scheduler Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Scheduler server model element index |
| 6 | uint8 | transition_time | Transition time for this action |
| 7-8 | uint16 | scene_number | Scene number being activated |

**C Functions**

```
/* Event id */
gecko_evt_mesh_scheduler_server_scene_changed_id

/* Event structure */
struct gecko_msg_mesh_scheduler_server_scene_changed_evt_t
{
  uint16 elem_index;,
  uint8 transition_time;,
  uint16 scene_number;
};
```

## 2.31 Bluetooth Mesh Sensor Client (mesh_sensor_client)

This class provides the commands and messages to interface with the Sensor Client model

### 2.31.1 mesh_sensor_client commands

#### 2.31.1.1 cmd_mesh_sensor_client_deinit

Deinitializes the Sensor Client model. There are no sensor-specific configurations to reset. Normally, models are initialized at boot and never deinitialized.

**Table 2.1001. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x01 | method | Message ID |

**Table 2.1002. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_sensor_client_deinit_rsp_t *gecko_cmd_mesh_sensor_client_deinit();

/* Response id */
gecko_rsp_mesh_sensor_client_deinit_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.2 cmd_mesh_sensor_client_get

Sends a Sensor Get message to fetch the Sensor Data state of one specific sensor given by its Property ID, results in a Sensor Status event. The Property ID 0x0000 can be used to fetch all sensor values present at a server element.

**Table 2.1003. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, or 0x0000 when not used to get values for all sensors present in the element. |

**Table 2.1004. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_sensor_client_get_rsp_t  *gecko_cmd_mesh_sensor_client_get(uint16  elem_index,  uint16
server_address, uint16 appkey_index, uint8 flags, uint16 property_id);

/* Response id */
gecko_rsp_mesh_sensor_client_get_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.3 cmd_mesh_sensor_client_get_cadence

Sends a Sensor Get Cadence message to get the Sensor Cadence state, which results in a Sensor Cadence Status message.

**Table 2.1005. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |

**Table 2.1006. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_client_get_cadence_rsp_t    *gecko_cmd_mesh_sensor_client_get_cadence(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id);

/* Response id */
gecko_rsp_mesh_sensor_client_get_cadence_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_cadence_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.4 cmd_mesh_sensor_client_get_column

Get a Sensor Series Column state, results in a Sensor Column Status event.

**Table 2.1007. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | column_id | Raw value identifying a column |

**Table 2.1008. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_client_get_column_rsp_t    *gecko_cmd_mesh_sensor_client_get_column(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id, uint8 column_id_len,
const uint8 *column_id_data);

/* Response id */
gecko_rsp_mesh_sensor_client_get_column_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_column_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.5 cmd_mesh_sensor_client_get_descriptor

Get the Sensor Descriptor state of one specific sensor or all sensors within a model. Results in a Sensor Descriptor Status event

**Table 2.1009. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | ProperyID for the sensor (optional). Range: 0x0001 - 0xffff for a specific device property ID or 0x0000 to get all (the value 0x0000 is prohibited as a real ID). |

**Table 2.1010. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_sensor_client_get_descriptor_rsp_t  *gecko_cmd_mesh_sensor_client_get_descriptor(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id);

/* Response id */
gecko_rsp_mesh_sensor_client_get_descriptor_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_descriptor_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.6 cmd_mesh_sensor_client_get_series

Get a Sensor Series Column state, which results in a Sensor Series Status event.

**Table 2.1011. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | column_ids | Raw values identifying starting and ending columns |

**Table 2.1012. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_client_get_series_rsp_t    *gecko_cmd_mesh_sensor_client_get_series(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id, uint8 column_ids_len,
const uint8 *column_ids_data);

/* Response id */
gecko_rsp_mesh_sensor_client_get_series_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_series_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.7 cmd_mesh_sensor_client_get_setting

Sends a Sensor Get Setting message to get the value of a specific setting for the given sensor, which results in a Sensor Setting Status event.

**Table 2.1013. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13-14 | uint16 | setting_id | Sensor Setting Property ID field identifying the device property of a setting. Range: 0x0001 - 0xffff, 0x0000 is prohibited. |

**Table 2.1014. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_client_get_setting_rsp_t    *gecko_cmd_mesh_sensor_client_get_setting(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id, uint16 setting_id);

/* Response id */
gecko_rsp_mesh_sensor_client_get_setting_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_setting_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.8 cmd_mesh_sensor_client_get_settings

Sends a Sensor Settings Get message to fetch the Sensor Property IDs present for the given sensor, which results in a Sensor Settings Status event.

**Table 2.1015. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |

**Table 2.1016. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_sensor_client_get_settings_rsp_t   *gecko_cmd_mesh_sensor_client_get_settings(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id);

/* Response id */
gecko_rsp_mesh_sensor_client_get_settings_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_get_settings_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.9 cmd_mesh_sensor_client_init

Initializes the Sensor Client model. Sensor Client does not have any internal configuration, it only activates the model in the Bluetooth mesh stack.

**Table 2.1017. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x00 | method | Message ID |

**Table 2.1018. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_sensor_client_init_rsp_t *gecko_cmd_mesh_sensor_client_init();

/* Response id */
gecko_rsp_mesh_sensor_client_init_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_init_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.10 cmd_mesh_sensor_client_set_cadence

Sends a Sensor Cadence Set message, either acknowledged or unacknowledged, depending on the message flags. Acknowledged message results in a Cadence Status reply message and event. The server must publish its new state in any case.

**Table 2.1019. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET CADENCE message will be sent, zero will send SET CADENCE UNACKNOWLEDGED |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | params | Byte array containing serialized fields of Sensor Cadence state, excluding the property ID<br>• Fast Cadence Period Divisor, 7 bits<br>• Status Trigger type, 1 bit (0 = discrete value, 1 = percentage)<br>• Status Trigger Delta Down, variable length<br>• Status Trigger Delta Up, variable length<br>• Status Min Interval, 8 bits, representing a power of 2 milliseconds. Valid range is 0-26<br>• Fast Cadence Low, variable length, lower bound for the fast cadence range<br>• Low Cadence Low, variable length, higher bound for the fast cadence range |

**Table 2.1020. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_client_set_cadence_rsp_t    *gecko_cmd_mesh_sensor_client_set_cadence(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id, uint8 params_len,
const uint8 *params_data);

/* Response id */
gecko_rsp_mesh_sensor_client_set_cadence_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_set_cadence_rsp_t
{
  uint16 result;
};
```

#### 2.31.1.11 cmd_mesh_sensor_client_set_setting

Sends Sensor Setting Set message to update the value of a specific setting for the given sensor, either acknowledged or unacknowledged depending on the message flags. Only acknowledged requests will have a direct Sensor Setting Status reply. The server must publish its new state in any case.

**Table 2.1021. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, SET SETTING message is sent, zero will use SET SETTING UNACKNOWLEDGED. |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13-14 | uint16 | setting_id | Sensor Setting Property ID field identifying the device property of a setting. Range: 0x0001 - 0xffff, 0x0000 is prohibited. |
| 15 | uint8array | raw_value | Sensor Setting raw value. Size and representation depends on the type defined by the Sensor Setting Property ID. |

**Table 2.1022. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_client_set_setting_rsp_t    *gecko_cmd_mesh_sensor_client_set_setting(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 flags, uint16 property_id, uint16 setting_id,
uint8 raw_value_len, const uint8 *raw_value_data);

/* Response id */
gecko_rsp_mesh_sensor_client_set_setting_id

/* Response structure */
struct gecko_msg_mesh_sensor_client_set_setting_rsp_t
```

```
{
  uint16 result;
};
```

### 2.31.2 mesh_sensor_client events

#### 2.31.2.1 evt_mesh_sensor_client_cadence_status

Indicates an incoming Sensor Cadence Status message.

**Table 2.1023. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use. |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8array | params | Optional byte array containing serialized fields of Sensor Cadence state, excluding the property ID<br>• Fast Cadence Period Divisor, 7 bits<br>• Status Trigger type, 1 bit (0 = discrete value, 1 = percentage)<br>• Status Trigger Delta Down, variable length<br>• Status Trigger Delta Up, variable length<br>• Status Min Interval, 8 bits, representing a power of 2 milliseconds. Valid range is 0-26<br>• Fast Cadence Low, variable length, lower bound for the fast cadence range<br>• Low Cadence Low, variable length, higher bound for the fast cadence range |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_cadence_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_cadence_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint8array params;
};
```

#### 2.31.2.2 evt_mesh_sensor_client_column_status

Indicates an incoming Sensor Column Status event.

**Table 2.1024. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8array | sensor_data | Byte array containing the serialized Sensor Series Column state in the following format:<br>• Sensor Raw Value X, variable length raw value representing the left corner of a column<br>• Sensor Column Width, variable length raw value representing the width of the column<br>• Sensor Raw Value Y, variable length raw value representing the height of the column<br>If the requested Property ID or column ID does not exist at the server elements, the reply status message contains only these two fields, omitting column width and height values. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_column_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_column_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint8array sensor_data;
};
```

#### 2.31.2.3 evt_mesh_sensor_client_descriptor_status

Indicates an incoming Sensor Descriptor Status message.

**Table 2.1025. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use. |
| 12 | uint8 | flags | No flags defined currently |
| 13 | uint8array | descriptors | One or more Sensor Descriptor states (N times 8 bytes) As a reply to a Get message referencing a sensor that does not exist, the array contains only the requested Property ID. Format of the Descriptor state is as follows:<br>• Property ID, 16 bits<br>• Sensor Positive Tolerance, 12 bits<br>• Sensor Negative Tolerance, 12 bits<br>• Sensor Sampling Function, 8 bits<br>• Sensor Measurement Period, 8 bits<br>• Sensor Update Interval, 8 bits |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_descriptor_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_descriptor_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint8array descriptors;
};
```

#### 2.31.2.4 evt_mesh_sensor_client_publish

Indicates that the publishing period timer elapsed and the app should/can publish its state or any request.

**Table 2.1026. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-9 | uint32 | period_ms | The current publishing period that can be used to estimate the next tick, e.g., when the state should be reported at higher frequency. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_publish_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_publish_evt_t
{
  uint16 elem_index;,
  uint32 period_ms;
};
```

#### 2.31.2.5 evt_mesh_sensor_client_series_status

Indicates an incoming Sensor Series Status message.

**Table 2.1027. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8array | sensor_data | Byte array containing the serialized sequence of Sensor Serier Column states in the following format:<br>• 1st Sensor Raw Value X, variable length raw value representing the left corner of a column<br>• 1st Sensor Column Width, variable length raw value representing the width of the column<br>• 1st Sensor Raw Value Y, variable length raw value representing the height of the column<br>• ...<br>• Nth Sensor Raw Value X, variable length raw value representing the left corner of a column<br>• Nth Sensor Column Width, variable length raw value representing the width of the column<br>• Nth Sensor Raw Value Y, variable length raw value representing the height of the column<br>If a Propery ID or column requested does not exist at the server element, the reply Series Status message contains only the given Property ID. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_series_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_series_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint8array sensor_data;
};
```

#### 2.31.2.6 evt_mesh_sensor_client_setting_status

Indicates an incoming Sensor Setting Status message.

**Table 2.1028. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15-16 | uint16 | setting_id | Sensor Setting Property ID field identifying the device property of a setting. Range: 0x0001 - 0xffff, 0x0000 is prohibited. |
| 17 | uint8array | raw_value | Sensor Setting raw value. Size and representation depends on the type defined by the Sensor Setting Property ID. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_setting_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_setting_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint16 setting_id;,
  uint8array raw_value;
};
```

#### 2.31.2.7 evt_mesh_sensor_client_settings_status

Indicates an incoming Sensor Settings Status message.

**Table 2.1029. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8array | setting_ids | Array of 16-bit Setting Property IDs of the settings the given sensor has |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_settings_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_settings_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint8array setting_ids;
};
```

#### 2.31.2.8 evt_mesh_sensor_client_status

Indicates an incoming Sensor Status event as a published data state from the sensor server or as a reply to Sensor Get request.

**Table 2.1030. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x49 | class | Message class: Bluetooth Mesh Sensor Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Source server model address |
| 8-9 | uint16 | client_address | Destination client model address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13 | uint8array | sensor_data | Serialized Sensor Data consisting of one or more Sensor state for each sensor within the element. To simplify processing, the byte array is in TLV format:<br>• 1st Property ID: 16 bits<br>• Value Length: 8 bits<br>• Value: variable<br>• 2nd Property ID: 16 bits<br>• Value Length: 8 bits<br>• Value: variable<br>• ...<br>If the requested Property ID does not exist at the server element, the reply contains only the given Property ID and zero length. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_client_status_id

/* Event structure */
struct gecko_msg_mesh_sensor_client_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint8array sensor_data;
};
```

## 2.32 Bluetooth Mesh Sensor Server Model (mesh_sensor_server)

This class provides the commands and messages to interface with the Sensor Server model.

A Sensor State consists of four states:

- Sensor Descriptor
- Sensor Setting
- Sensor Cadence
- Measurement value

A multisensor setup is possible by having multiple sensor states within the same model, provided that each sensor has a unique Sensor Property ID.

Sensor Descriptor states are constant. Therefore, the stack can cache them and enumerate the present sensors to clients when requested.

Currently, the Sensor Server model does not cache the measurement data, sensor settings, or cadence. When a client is querying sensor data, the requests will be propagated to the application.

### 2.32.1 mesh_sensor_server commands

#### 2.32.1.1 cmd_mesh_sensor_server_deinit

Deinitializes the Sensor Server functionality. Note that the heap reserved space cannot be freed or reallocated. Reinitializing with greater number of sensors than before will eventually return an out of memory error until the device is reset.

**Table 2.1031. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1032. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_sensor_server_deinit_rsp_t *gecko_cmd_mesh_sensor_server_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_sensor_server_deinit_id

/* Response structure */
struct gecko_msg_mesh_sensor_server_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.32.1.2 cmd_mesh_sensor_server_init

Initializes the Sensor Server model with Sensor Descriptors present at the element. The descriptors are cached and Descriptor Get queries are served without propagating it to the application. All incoming client queries are checked against the cached data, only valid requests related to existing sensors are propagated to the the application.

**Table 2.1033. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6 | uint8array | descriptors | Sensor Descriptor State structures submitted as a byte array<br>A sensor descriptor represents the attributes describing the sensor data. It does not change throughout the lifetime of the element. The following fields are required:<br>• Sensor Property ID: 16 bits<br>• Sensor Positive Tolerance: 12 bits<br>• Sensor Negative Tolerance: 12 bits<br>• Sensor Sampling Function: 8 bits<br>• Sensor Measurement Period: 8 bits<br>• Sensor Update Interval: 8 bits<br>Sensor Property ID is a 2-octet value referencing a device property that describes the meaning and the format of data reported by the sensor. The value 0x0000 is prohibited. Valid range is 0x0001 to 0xFFFF. |

**Table 2.1034. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_sensor_server_init_rsp_t  *gecko_cmd_mesh_sensor_server_init(uint16  elem_index,  uint8
descriptors_len, const uint8 *descriptors_data);

/* Response id */
gecko_rsp_mesh_sensor_server_init_id

/* Response structure */
struct gecko_msg_mesh_sensor_server_init_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.32.1.3 cmd_mesh_sensor_server_send_column_status

Send Column Status message as a response to a Column Get client request or as an unsolicited message

**Table 2.1035. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | client_address | Destination client address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | sensor_data | Byte array containing the serialized Sensor Series Column state in the following format:<br>• Sensor Raw Value X, variable length raw value representing the left corner of a column<br>• Sensor Column Width, variable length raw value representing the width of the column<br>• Sensor Raw Value Y, variable length raw value representing the height of the column<br>If the Property ID or the column ID (Raw value X) does not exist, the reply must contain only these two fields, omitting the optional Column Width and Raw Value Y fields. |

**Table 2.1036. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_sensor_server_send_column_status_rsp_t
*gecko_cmd_mesh_sensor_server_send_column_status(uint16    elem_index,    uint16    client_address,    uint16
appkey_index, uint8 flags, uint16 property_id, uint8 sensor_data_len, const uint8 *sensor_data_data);

/* Response id */
gecko_rsp_mesh_sensor_server_send_column_status_id
```

```
/* Response structure */
struct gecko_msg_mesh_sensor_server_send_column_status_rsp_t
{
  uint16 result;
};
```

#### 2.32.1.4 cmd_mesh_sensor_server_send_descriptor_status

Send a Descriptor Status message either as a reply to a Get Descriptor client request.

**Table 2.1037. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | uint16 | client_address | Destination client address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11 | uint8array | descriptors | Serialized Sensor Descriptor states for all sensors within the element consisting one or more 8 bytes structures as follows:<br>• Sensor Property ID: 16 bits<br>• Sensor Positive Tolerance: 12 bits<br>• Sensor Negative Tolerance: 12 bits<br>• Sensor Sampling Function: 8 bits<br>• Sensor Measurement Period: 8 bits |

**Table 2.1038. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_sensor_server_send_descriptor_status_rsp_t
*gecko_cmd_mesh_sensor_server_send_descriptor_status(uint16   elem_index,   uint16   client_address,   uint16
appkey_index, uint8 flags, uint8 descriptors_len, const uint8 *descriptors_data);

/* Response id */
gecko_rsp_mesh_sensor_server_send_descriptor_status_id

/* Response structure */
struct gecko_msg_mesh_sensor_server_send_descriptor_status_rsp_t
{
  uint16 result;
};
```

#### 2.32.1.5 cmd_mesh_sensor_server_send_series_status

Send Series Status message as a response to a Series Get client request or as an unsolicited message

**Table 2.1039. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | client_address | Destination client address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | sensor_data | Byte array containing the serialized sequence of Sensor Series Column states in the following format:<br>• 1st Sensor Raw Value X, variable length raw value representing the left corner of a column<br>• 1st Sensor Column Width, variable length raw value representing the width of the column<br>• 1st Sensor Raw Value Y, variable length raw value representing the height of the column<br>• ...<br>• Nth Sensor Raw Value X, variable length raw value representing the left corner of a column<br>• Nth Sensor Column Width, variable length raw value representing the width of the column<br>• Nth Sensor Raw Value Y, variable length raw value representing the height of the column<br>If Property ID does not exist in the element, the reply must contain only the given Property ID, omitting the other optional fields to column identifiers and column values. |

**Table 2.1040. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_sensor_server_send_series_status_rsp_t
*gecko_cmd_mesh_sensor_server_send_series_status(uint16    elem_index,    uint16    client_address,    uint16
appkey_index, uint8 flags, uint16 property_id, uint8 sensor_data_len, const uint8 *sensor_data_data);

/* Response id */
gecko_rsp_mesh_sensor_server_send_series_status_id

/* Response structure */
struct gecko_msg_mesh_sensor_server_send_series_status_rsp_t
{
  uint16 result;
};
```

#### 2.32.1.6 cmd_mesh_sensor_server_send_status

Send Sensor Status message as a reply to a Get client request or as an unsolicited message.

**Table 2.1041. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Setup Server model element index |
| 6-7 | uint16 | client_address | Destination client address. The address 0x0000 can be used to publish the message according to model configuration |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | No flags defined currently |
| 11 | uint8array | sensor_data | Serialized Sensor Data consisting of one or more Sensor state for each sensor within the element. To simplify processing, the byte array is in TLV format:<br>• 1st Property ID: 16 bits<br>• Value Length: 8 bits<br>• Value: variable<br>• 2nd Property ID: 16 bits<br>• Value Length: 8 bits<br>• Value: variable<br>• ...<br>If sensor data was requested for a Property ID that does not exist within the element, the reply must contain the given Property ID with zero length. |

**Table 2.1042. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_sensor_server_send_status_rsp_t    *gecko_cmd_mesh_sensor_server_send_status(uint16
elem_index,  uint16  client_address,  uint16  appkey_index,  uint8  flags,  uint8  sensor_data_len,  const  uint8
*sensor_data_data);

/* Response id */
gecko_rsp_mesh_sensor_server_send_status_id
```

```
/* Response structure */
struct gecko_msg_mesh_sensor_server_send_status_rsp_t
{
  uint16 result;
};
```

### 2.32.2 mesh_sensor_server events

#### 2.32.2.1 evt_mesh_sensor_server_get_column_request

Indicates an incoming Sensor Column Get message to get Sensor Series Column state. The event must be replied to by sending a Sensor Column Status message.

**Table 2.1043. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | uint16 | client_address | Source client model address |
| 8-9 | uint16 | server_address | Destination server address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8array | column_ids | Raw value identifying a column |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_server_get_column_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_server_get_column_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint8array column_ids;
};
```

#### 2.32.2.2 evt_mesh_sensor_server_get_request

Indicates an incoming Sensor Get request to get the Sensor Data state(s). This event must be replied to by sending the Sensor Status message.

**Table 2.1044. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | uint16 | client_address | Source client address |
| 8-9 | uint16 | server_address | Destination server address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, or 0x0000 to get the values of all sensors. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_server_get_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_server_get_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;
};
```

#### 2.32.2.3 evt_mesh_sensor_server_get_series_request

Indicates an incoming Sensor Series Get message to get the Sensor Series Column states. This event must be replied to by sending a Sensor Series Status message.

**Table 2.1045. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | uint16 | client_address | Source client address |
| 8-9 | uint16 | server_address | Destination server address |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8array | column_ids | Optional raw values identifying starting and ending columns |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_server_get_series_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_server_get_series_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint8array column_ids;
};
```

#### 2.32.2.4 evt_mesh_sensor_server_publish

Indicates that the publishing period timer elapsed and the app should publish its state.

**Table 2.1046. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x47 | class | Message class: Bluetooth Mesh Sensor Server Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-9 | uint32 | period_ms | The current publishing period that can be used to estimate the next tick, e.g., when the state should be reported at higher frequency. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_server_publish_id

/* Event structure */
struct gecko_msg_mesh_sensor_server_publish_evt_t
{
  uint16 elem_index;,
  uint32 period_ms;
};
```

## 2.33 Bluetooth Mesh Sensor Setup Server (mesh_sensor_setup_server)

This class provides the commands and messages to interface with the Sensor Setup Server model. Elements containing sensor model must have a setup server model attached. Therefore, it is initialized/deinitialized internally together with the server model.

### 2.33.1 mesh_sensor_setup_server commands

#### 2.33.1.1 cmd_mesh_sensor_setup_server_send_cadence_status

Replies to a Get/Set Cadence client request with a Cadence Status message. Only Cadence Set (acknowledged) must be answered by sending the status message to the client. In addition, configuration changes must be published according to model publishing configuration.

**Table 2.1047. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | client_address | Destination client address The address 0x0000 can be used to publish the message according model configuration instead of a direct reply. |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | params | Optional byte array containing the serialized Sensor Cadence state, excluding the property ID. If not empty, the state consists of the following fields:<br>• Fast Cadence Period Divisor, 7 bits<br>• Status Trigger type, 1 bits (0 = discrete value, 1 = percentage)<br>• Status Trigger Delta Down, variable length<br>• Status Trigger Delta Up, variable length<br>• Status Min Interval, 8 bits, representing a power of 2 milliseconds. Valid range is 0-26<br>• Fast Cadence Low, variable length, lower bound for the fast cadence range<br>• Low Cadence Low, variable length, higher bound for the fast cadence range |

**Table 2.1048. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_sensor_setup_server_send_cadence_status_rsp_t
*gecko_cmd_mesh_sensor_setup_server_send_cadence_status(uint16  elem_index,  uint16  client_address,  uint16
appkey_index, uint8 flags, uint16 property_id, uint8 params_len, const uint8 *params_data);

/* Response id */
gecko_rsp_mesh_sensor_setup_server_send_cadence_status_id

/* Response structure */
struct gecko_msg_mesh_sensor_setup_server_send_cadence_status_rsp_t
{
  uint16 result;
};
```

#### 2.33.1.2 cmd_mesh_sensor_setup_server_send_setting_status

Replies to a Get/Set Setting client request with a Setting Status message. Only Set Setting (acknowledged) request must be answered by sending a reply to the unicast address of the sender. In addition, configuration changes must be published if model publishing is set up.

**Table 2.1049. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | client_address | Destination client model address |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13-14 | uint16 | setting_id | Sensor Setting Property ID field identifying the device property of a setting. Range: 0x0001 - 0xffff, 0x0000 is prohibited. |
| 15 | uint8array | raw_value | Sensor Setting raw value. Size and representation depends on the type defined by the Sensor Setting Property ID. |

**Table 2.1050. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_sensor_setup_server_send_setting_status_rsp_t
*gecko_cmd_mesh_sensor_setup_server_send_setting_status(uint16  elem_index,  uint16  client_address,  uint16
appkey_index,  uint8  flags,  uint16  property_id,  uint16  setting_id,  uint8  raw_value_len,  const  uint8
*raw_value_data);

/* Response id */
gecko_rsp_mesh_sensor_setup_server_send_setting_status_id

/* Response structure */
struct gecko_msg_mesh_sensor_setup_server_send_setting_status_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.33.1.3 cmd_mesh_sensor_setup_server_send_settings_status

Replies to a Get Settings client request with a Settings Status message.

**Table 2.1051. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | client_address | Destination client model address |
| 8-9 | uint16 | appkey_index | The application key index to use |
| 10 | uint8 | flags | No flags defined currently |
| 11-12 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 13 | uint8array | setting_ids | Array of 16-bit Setting Property IDs of the settings the given sensor has |

**Table 2.1052. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_sensor_setup_server_send_settings_status_rsp_t
*gecko_cmd_mesh_sensor_setup_server_send_settings_status(uint16  elem_index,  uint16  client_address,  uint16
appkey_index, uint8 flags, uint16 property_id, uint8 setting_ids_len, const uint8 *setting_ids_data);

/* Response id */
gecko_rsp_mesh_sensor_setup_server_send_settings_status_id

/* Response structure */
struct gecko_msg_mesh_sensor_setup_server_send_settings_status_rsp_t
{
  uint16 result;
};
```

### 2.33.2 mesh_sensor_setup_server events

#### 2.33.2.1 evt_mesh_sensor_setup_server_get_cadence_request

Indicates an incoming Sensor Cadence Get request. This event must be replied to by sending a Sensor Cadence Status message.

**Table 2.1053. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Setup Server model element index |
| 6-7 | uint16 | client_address | Requesting client model's address |
| 8-9 | uint16 | server_address | Address the request was directed to, either the server's unicast address or a group address the server subscribes to |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_setup_server_get_cadence_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_setup_server_get_cadence_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;
};
```

#### 2.33.2.2 evt_mesh_sensor_setup_server_get_setting_request

Indicates an incoming Sensor Get Setting request to fetch the value of a setting to a given sensor of a setting given by its ID. This event must be replied to by sending a Sensor Setting Status message.

**Table 2.1054. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Setup Server model element index |
| 6-7 | uint16 | client_address | Requesting client model's address |
| 8-9 | uint16 | server_address | Address the request was directed to, either the server's unicast address, or a group address the server subscribes to |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15-16 | uint16 | setting_id | Sensor Setting Property ID field identifying the device property of a setting. Range: 0x0001 - 0xffff, 0x0000 is prohibited. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_setup_server_get_setting_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_setup_server_get_setting_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint16 setting_id;
};
```

#### 2.33.2.3 evt_mesh_sensor_setup_server_get_settings_request

Indicates an incoming Sensor Settings Get message to fetch the Sensor Setting Property IDs configured for the given Sensor. This event must be replied to by sending a Sensor Settings Status message.

**Table 2.1055. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Setup Server model element index |
| 6-7 | uint16 | client_address | Requesting client model's address |
| 8-9 | uint16 | server_address | Address the request was directed to, either the server's unicast address or a group address the server subscribes to |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | No flags defined currently |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_setup_server_get_settings_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_setup_server_get_settings_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;
};
```

#### 2.33.2.4 evt_mesh_sensor_setup_server_publish

Indicates that the publishing period timer elapsed and the app should/can publish its state

**Table 2.1056. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-9 | uint32 | period_ms | The current publishing period that can be used to estimate the next tick, e.g., when the state should be reported at higher frequency. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_setup_server_publish_id

/* Event structure */
struct gecko_msg_mesh_sensor_setup_server_publish_evt_t
{
  uint16 elem_index;,
  uint32 period_ms;
};
```

#### 2.33.2.5 evt_mesh_sensor_setup_server_set_cadence_request

Indicates an incoming Sensor Cadence Set request, which can be replied to by sending a Sensor Cadence Status message. Only Sensor Cadence Set (acknowledged) request results in a direct reply. In addition, configuration changes must be reported by publishing the updated cadence state according to model configuration.

**Table 2.1057. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Setup Server model element index |
| 6-7 | uint16 | client_address | Requesting client model's address |
| 8-9 | uint16 | server_address | Address the request was directed to, either the server's unicast address or a group address the server subscribes to |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, the client sent the message with SET CADENCE opcode and expects a CADENCE STATUS message in return. |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15 | uint8 | period_divisor | Fast Cadence Period Divisor, 7 bits defining the divisor for the Publish Period |
| 16 | uint8 | trigger_type | Status Trigger Type, 1 bit: 0 = discrete value, 1 = delta percentage |
| 17 | uint8array | params | Optional byte array containing serialized fields of Sensor Cadence state, excluding the property ID, period divisor and trigger type<br>• Fast Cadence Period Divisor, 7 bits<br>• Status Trigger type, 1 bit (0 = discrete value, 1 = percentage)<br>• Status Trigger Delta Down, variable length<br>• Status Trigger Delta Up, variable length<br>• Status Min Interval, 8 bits, representing a power of 2 milliseconds. Valid range is 0-26<br>• Fast Cadence Low, variable length, lower bound for the fast cadence range<br>• Low Cadence Low, variable length, higher bound for the fast cadence range |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_setup_server_set_cadence_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_setup_server_set_cadence_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
```

```
  uint8 period_divisor;,
  uint8 trigger_type;,
  uint8array params;
};
```

#### 2.33.2.6 evt_mesh_sensor_setup_server_set_setting_request

Indicates an incoming Sensor Set Setting request, which can be replied to by sending a Sensor Setting Status message. Only Setting Set (acknowledged) request is replied directly to the client. In addition, configuration changes must be reported by publishing the new state according to model publishing configuration.

**Table 2.1058. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x48 | class | Message class: Bluetooth Mesh Sensor Setup Server |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Setup Server model element index |
| 6-7 | uint16 | client_address | Requesting client address |
| 8-9 | uint16 | server_address | Address the request was directed to, either the server's unicast address, or a group address the server subscribes to. |
| 10-11 | uint16 | appkey_index | The application key index to use |
| 12 | uint8 | flags | Bit 1 (0x02) defines whether response is required. If set to 1, the client sent the message with SET SETTING opcode and expects a SETTING STATUS message in return. |
| 13-14 | uint16 | property_id | Property ID for the sensor. Range: 0x0001 - 0x0ffff for a specific device property, the value 0x0000 is prohibited. |
| 15-16 | uint16 | setting_id | Sensor Setting Property ID field identifying the device property of a setting. Range: 0x0001 - 0xffff, 0x0000 is prohibited. |
| 17 | uint8array | raw_value | Sensor Setting raw value. Size and representation depends on the type defined by the Sensor Setting Property ID. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_sensor_setup_server_set_setting_request_id

/* Event structure */
struct gecko_msg_mesh_sensor_setup_server_set_setting_request_evt_t
{
  uint16 elem_index;,
  uint16 client_address;,
  uint16 server_address;,
  uint16 appkey_index;,
  uint8 flags;,
  uint16 property_id;,
  uint16 setting_id;,
  uint8array raw_value;
};
```

## 2.34 Bluetooth Mesh Test Utilities (mesh_test)

These commands are meant for development and testing. Do not use in production software.

### 2.34.1 mesh_test commands

#### 2.34.1.1 cmd_mesh_test_add_local_key

Add a network or application key locally.

**Table 2.1059. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x15 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1a | method | Message ID |
| 4 | uint8 | key_type | 0 for network key, 1 for application key |
| 5-20 | aes_key_128 | key | Key data |
| 21-22 | uint16 | key_index | Index for the added key (must be unused) |
| 23-24 | uint16 | netkey_index | Network key index to which the application key is bound; ignored for network keys |

**Table 2.1060. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_add_local_key_rsp_t *gecko_cmd_mesh_test_add_local_key(uint8 key_type, aes_key_128
key, uint16 key_index, uint16 netkey_index);

/* Response id */
gecko_rsp_mesh_test_add_local_key_id

/* Response structure */
struct gecko_msg_mesh_test_add_local_key_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.2 cmd_mesh_test_add_local_model_sub

Add an address to a local model's subscription list.

**Table 2.1061. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |
| 10-11 | uint16 | sub_address | The address to add to the subscription list |

**Table 2.1062. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_add_local_model_sub_rsp_t    *gecko_cmd_mesh_test_add_local_model_sub(uint16
elem_index, uint16 vendor_id, uint16 model_id, uint16 sub_address);

/* Response id */
gecko_rsp_mesh_test_add_local_model_sub_id

/* Response structure */
struct gecko_msg_mesh_test_add_local_model_sub_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.3 cmd_mesh_test_add_local_model_sub_va

Add a virtual address to a local model's subscription list.

**Table 2.1063. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |
| 10 | uint8array | sub_address | The Label UUID to add to the subscription list. The array must be exactly 16 bytes long. |

**Table 2.1064. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_add_local_model_sub_va_rsp_t  *gecko_cmd_mesh_test_add_local_model_sub_va(uint16
elem_index, uint16 vendor_id, uint16 model_id, uint8 sub_address_len, const uint8 *sub_address_data);

/* Response id */
gecko_rsp_mesh_test_add_local_model_sub_va_id

/* Response structure */
struct gecko_msg_mesh_test_add_local_model_sub_va_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.4 cmd_mesh_test_bind_local_model_app

Bind a Model to an Appkey locally.

**Table 2.1065. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target Element, 0 is primary element |
| 6-7 | uint16 | appkey_index | The Appkey to use for binding |
| 8-9 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for SIG models. |
| 10-11 | uint16 | model_id | Model ID |

**Table 2.1066. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_bind_local_model_app_rsp_t    *gecko_cmd_mesh_test_bind_local_model_app(uint16
elem_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_test_bind_local_model_app_id

/* Response structure */
struct gecko_msg_mesh_test_bind_local_model_app_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.5 cmd_mesh_test_cancel_segmented_tx

Cancel sending a segmented message.

**Table 2.1067. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | src_addr | Source address for the segmented message |
| 6-7 | uint16 | dst_addr | Destination address for the segmented message |

**Table 2.1068. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x25 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_cancel_segmented_tx_rsp_t *gecko_cmd_mesh_test_cancel_segmented_tx(uint16 src_addr,
uint16 dst_addr);

/* Response id */
gecko_rsp_mesh_test_cancel_segmented_tx_id

/* Response structure */
struct gecko_msg_mesh_test_cancel_segmented_tx_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.6 cmd_mesh_test_clear_replay_protection_list_entry

Clear replay protection list entry for an address. This command needs to be used with care, as it may expose the node to replay attacks when wrongly used.

**Table 2.1069. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x2b | method | Message ID |
| 4-5 | uint16 | address | Source address to use in finding the entry |

**Table 2.1070. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x2b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_test_clear_replay_protection_list_entry_rsp_t
*gecko_cmd_mesh_test_clear_replay_protection_list_entry(uint16 address);

/* Response id */
gecko_rsp_mesh_test_clear_replay_protection_list_entry_id

/* Response structure */
struct gecko_msg_mesh_test_clear_replay_protection_list_entry_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.7 cmd_mesh_test_del_local_key

Delete a network or application key locally.

**Table 2.1071. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1b | method | Message ID |
| 4 | uint8 | key_type | 0 for network key, 1 for application key |
| 5-6 | uint16 | key_index | Index of the key to delete |

**Table 2.1072. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_del_local_key_rsp_t  *gecko_cmd_mesh_test_del_local_key(uint8  key_type,  uint16
key_index);

/* Response id */
gecko_rsp_mesh_test_del_local_key_id

/* Response structure */
struct gecko_msg_mesh_test_del_local_key_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.8 cmd_mesh_test_del_local_model_sub

Remove an address from a local model's subscription list.

**Table 2.1073. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |
| 10-11 | uint16 | sub_address | The address to remove from the subscription list |

**Table 2.1074. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_del_local_model_sub_rsp_t    *gecko_cmd_mesh_test_del_local_model_sub(uint16
elem_index, uint16 vendor_id, uint16 model_id, uint16 sub_address);

/* Response id */
gecko_rsp_mesh_test_del_local_model_sub_id

/* Response structure */
struct gecko_msg_mesh_test_del_local_model_sub_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.9 cmd_mesh_test_del_local_model_sub_va

Remove a virtual address from a local model's subscription list.

**Table 2.1075. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |
| 10 | uint8array | sub_address | The Label UUID to remove from the subscription list. The array must be exactly 16 bytes long. |

**Table 2.1076. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_del_local_model_sub_va_rsp_t  *gecko_cmd_mesh_test_del_local_model_sub_va(uint16
elem_index, uint16 vendor_id, uint16 model_id, uint8 sub_address_len, const uint8 *sub_address_data);

/* Response id */
gecko_rsp_mesh_test_del_local_model_sub_va_id

/* Response structure */
struct gecko_msg_mesh_test_del_local_model_sub_va_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.10 cmd_mesh_test_get_element_seqnum

Get current sequence number of an element.

**Table 2.1077. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1e | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |

**Table 2.1078. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | seqnum | Current sequence number of the element Ignore the value if the result code indicates an error (for example, when the element index is out of bounds). |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_get_element_seqnum_rsp_t *gecko_cmd_mesh_test_get_element_seqnum(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_test_get_element_seqnum_id

/* Response structure */
struct gecko_msg_mesh_test_get_element_seqnum_rsp_t
{
  uint16 result;,
  uint32 seqnum;
};
```

#### 2.34.1.11 cmd_mesh_test_get_ivupdate_test_mode

Get the current IV update test mode. See set IV update test mode for details.

**Table 2.1079. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x06 | method | Message ID |

**Table 2.1080. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | mode | Indicates whether test mode is enabled (1) or disabled (0). |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_get_ivupdate_test_mode_rsp_t *gecko_cmd_mesh_test_get_ivupdate_test_mode();

/* Response id */
gecko_rsp_mesh_test_get_ivupdate_test_mode_id

/* Response structure */
struct gecko_msg_mesh_test_get_ivupdate_test_mode_rsp_t
{
  uint16 result;,
  uint8 mode;
};
```

#### 2.34.1.12 cmd_mesh_test_get_key

Get key by position. Only current key data exits in normal mode. Old key data can be queried only during key refresh.

**Note:** this command handles sensitive data, and in Secure NCP applications requires encryption to be enabled. Otherwise the error bg_err_application_mismatched_or_insufficient_security will be returned.

**Table 2.1081. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x21 | method | Message ID |
| 4 | uint8 | type | 0 for network key, 1 for application key |
| 5-8 | uint32 | index | Key position, ranging from zero to key count minus one |
| 9 | uint8 | current | 1: Current key, 0: Old key |

**Table 2.1082. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x16 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x21 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | id | Mesh key index of the key |
| 8-9 | uint16 | network | For application keys, the network key index of the network key this key is bound to. Ignore for other key types. |
| 10-25 | aes_key_128 | key | Key data, 16 bytes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_get_key_rsp_t *gecko_cmd_mesh_test_get_key(uint8 type, uint32 index, uint8 current);

/* Response id */
gecko_rsp_mesh_test_get_key_id

/* Response structure */
struct gecko_msg_mesh_test_get_key_rsp_t
{
  uint16 result;,
  uint16 id;,
  uint16 network;,
  aes_key_128 key;
};
```

#### 2.34.1.13 cmd_mesh_test_get_key_count

Get total number of keys in node.

**Table 2.1083. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x20 | method | Message ID |
| 4 | uint8 | type | 0 for network key, 1 for application key |

**Table 2.1084. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x20 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | count | Number of keys of the given type on the device |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_get_key_count_rsp_t *gecko_cmd_mesh_test_get_key_count(uint8 type);

/* Response id */
gecko_rsp_mesh_test_get_key_count_id

/* Response structure */
struct gecko_msg_mesh_test_get_key_count_rsp_t
{
  uint16 result;,
  uint32 count;
};
```

#### 2.34.1.14 cmd_mesh_test_get_local_config

Get the value of a state in the Configuration Server model. Use this for testing and debugging purposes only.

**Table 2.1085. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | id | The state to read |
| 6-7 | uint16 | netkey_index | Network key index; ignored for node-wide states |

**Table 2.1086. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x19 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | data | Raw binary value |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_get_local_config_rsp_t  *gecko_cmd_mesh_test_get_local_config(uint16  id,  uint16
netkey_index);

/* Response id */
gecko_rsp_mesh_test_get_local_config_id

/* Response structure */
struct gecko_msg_mesh_test_get_local_config_rsp_t
{
  uint16 result;,
  uint8array data;
};
```

#### 2.34.1.15 cmd_mesh_test_get_local_heartbeat_publication

Get heartbeat publication state of a local node.

**Table 2.1087. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x16 | method | Message ID |

**Table 2.1088. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x16 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | publication_address | Heartbeat publication address |
| 8 | uint8 | count | Heartbeat publication remaining count |
| 9 | uint8 | period_log | Heartbeat publication period setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node will send a heartbeat message every 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |
| 10 | uint8 | ttl | Time-to-live parameter for heartbeat messages |
| 11-12 | uint16 | features | Heartbeat trigger setting |
| 13-14 | uint16 | publication_net-key_index | Index of the network key used to encrypt heartbeat messages |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_test_get_local_heartbeat_publication_rsp_t
*gecko_cmd_mesh_test_get_local_heartbeat_publication();

/* Response id */
gecko_rsp_mesh_test_get_local_heartbeat_publication_id

/* Response structure */
struct gecko_msg_mesh_test_get_local_heartbeat_publication_rsp_t
{
  uint16 result;,
  uint16 publication_address;,
  uint8 count;,
  uint8 period_log;,
  uint8 ttl;,
```

```
  uint16 features;,
  uint16 publication_netkey_index;
};
```

#### 2.34.1.16 cmd_mesh_test_get_local_heartbeat_subscription

Get the local node heartbeat subscription. state

**Table 2.1089. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x15 | method | Message ID |

**Table 2.1090. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x15 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | count | Number of received heartbeat messages |
| 8 | uint8 | hop_min | Minimum observed hop count in heartbeat messages |
| 9 | uint8 | hop_max | Maximum observed hop count in heartbeat messages |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_test_get_local_heartbeat_subscription_rsp_t
*gecko_cmd_mesh_test_get_local_heartbeat_subscription();

/* Response id */
gecko_rsp_mesh_test_get_local_heartbeat_subscription_id

/* Response structure */
struct gecko_msg_mesh_test_get_local_heartbeat_subscription_rsp_t
{
  uint16 result;,
  uint16 count;,
  uint8 hop_min;,
  uint8 hop_max;
};
```

#### 2.34.1.17 cmd_mesh_test_get_local_model_app_bindings

Get application key bindings of a model.

**Table 2.1091. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x29 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |

**Table 2.1092. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x29 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | appkeys | List of 16-bit application key indices; empty if model has not been bound to any application key. |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_test_get_local_model_app_bindings_rsp_t
*gecko_cmd_mesh_test_get_local_model_app_bindings(uint16 elem_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_test_get_local_model_app_bindings_id

/* Response structure */
struct gecko_msg_mesh_test_get_local_model_app_bindings_rsp_t
{
  uint16 result;,
  uint8array appkeys;
};
```

#### 2.34.1.18 cmd_mesh_test_get_local_model_pub

Get a local model's publication address, key, and parameters.

**Table 2.1093. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |

**Table 2.1094. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | appkey_index | The application key index used for the application messages published |
| 8-9 | uint16 | pub_address | The address published to |
| 10 | uint8 | ttl | Time-to-Live value for published messages |
| 11 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 12 | uint8 | retrans | See documentation of local model publication set command for details. |
| 13 | uint8 | credentials | Friendship credentials flag |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_get_local_model_pub_rsp_t    *gecko_cmd_mesh_test_get_local_model_pub(uint16
elem_index, uint16 vendor_id, uint16 model_id);
```

```
/* Response id */
gecko_rsp_mesh_test_get_local_model_pub_id

/* Response structure */
struct gecko_msg_mesh_test_get_local_model_pub_rsp_t
{
  uint16 result;,
  uint16 appkey_index;,
  uint16 pub_address;,
  uint8 ttl;,
  uint8 period;,
  uint8 retrans;,
  uint8 credentials;
};
```

#### 2.34.1.19 cmd_mesh_test_get_local_model_sub

Get all entries in a local model's subscription list.

**Table 2.1095. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 8-9 | uint16 | model_id | Model ID |

**Table 2.1096. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | addresses | List of 16-bit Mesh addresses; empty if not subscribed to any address. Ignore if result code is non-zero. |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_get_local_model_sub_rsp_t    *gecko_cmd_mesh_test_get_local_model_sub(uint16
elem_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_test_get_local_model_sub_id

/* Response structure */
struct gecko_msg_mesh_test_get_local_model_sub_rsp_t
{
  uint16 result;,
  uint8array addresses;
};
```

#### 2.34.1.20 cmd_mesh_test_get_nettx

Get the network transmit state of a node.

**Table 2.1097. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x00 | method | Message ID |

**Table 2.1098. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | count | Number of network layer transmissions beyond the initial one. Range: 0-7. |
| 7 | uint8 | interval | Transmit interval steps. The interval between transmissions is a random value between 10*(1+steps) and 10*(2+steps) milliseconds. For example, for a value of 2 the interval would be between 30 and 40 milliseconds. Range: 0-31. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_get_nettx_rsp_t *gecko_cmd_mesh_test_get_nettx();

/* Response id */
gecko_rsp_mesh_test_get_nettx_id

/* Response structure */
struct gecko_msg_mesh_test_get_nettx_rsp_t
{
  uint16 result;,
  uint8 count;,
  uint8 interval;
};
```

#### 2.34.1.21 cmd_mesh_test_get_relay

**Table 2.1099. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x02 | method | Message ID |

**Table 2.1100. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | enabled | State value indicating whether the relay functionality is not enabled on the node (0), is enabled on the node (1), or is not available (2). |
| 7 | uint8 | count | Number of relay transmissions beyond the initial one. Range: 0-7. |
| 8 | uint8 | interval | Relay retransmit interval steps. The interval between transmissions is 10*(1+steps) milliseconds. Range: 0-31. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_get_relay_rsp_t *gecko_cmd_mesh_test_get_relay();

/* Response id */
gecko_rsp_mesh_test_get_relay_id

/* Response structure */
struct gecko_msg_mesh_test_get_relay_rsp_t
{
  uint16 result;,
  uint8 enabled;,
  uint8 count;,
  uint8 interval;
};
```

#### 2.34.1.22 cmd_mesh_test_get_replay_protection_list_entry

Get replay protection list entry for an address.

**Table 2.1101. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x2a | method | Message ID |
| 4-5 | uint16 | address | Source address to check |

**Table 2.1102. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x2a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | uint32 | seq | Unsigned 32-bit integer |
| 10-13 | uint32 | seq_ivindex | Unsigned 32-bit integer |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_test_get_replay_protection_list_entry_rsp_t
*gecko_cmd_mesh_test_get_replay_protection_list_entry(uint16 address);

/* Response id */
gecko_rsp_mesh_test_get_replay_protection_list_entry_id

/* Response structure */
struct gecko_msg_mesh_test_get_replay_protection_list_entry_rsp_t
{
  uint16 result;,
  uint32 seq;,
  uint32 seq_ivindex;
};
```

#### 2.34.1.23 cmd_mesh_test_prov_get_device_key

Get device key by the address of the nodes primary element.

**Note:** this command handles sensitive data, and in Secure NCP applications requires encryption to be enabled. Otherwise the error bg_err_application_mismatched_or_insufficient_security will be returned.

**Table 2.1103. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x23 | method | Message ID |
| 4-5 | uint16 | address | Address of the node |

**Table 2.1104. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x23 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-21 | aes_key_128 | device_key | Device key, 16-bytes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_prov_get_device_key_rsp_t *gecko_cmd_mesh_test_prov_get_device_key(uint16 address);

/* Response id */
gecko_rsp_mesh_test_prov_get_device_key_id

/* Response structure */
struct gecko_msg_mesh_test_prov_get_device_key_rsp_t
{
  uint16 result;,
  aes_key_128 device_key;
};
```

#### 2.34.1.24 cmd_mesh_test_prov_prepare_key_refresh

Prepare key refresh by feeding the new network key and all needed application keys. The function can be called multiple times to include more application keys. The network key must be the same in all calls. If the network key is changed, the network key from the 1st command is used. Sending application key data with length zero results in all initialization data being forgotten unless this is done in the first prepare command i.e., trying to update only the network key. Also starting the key refresh procedure results in all the preparation data being forgotten.

**Note:** this command handles sensitive data, and in Secure NCP applications requires encryption to be enabled. Otherwise the error bg_err_application_mismatched_or_insufficient_security will be returned.

**Table 2.1105. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x24 | method | Message ID |
| 4-19 | aes_key_128 | net_key | New net key |
| 20 | uint8array | app_keys | list of new application keys, 16-bytes each |

**Table 2.1106. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x24 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_test_prov_prepare_key_refresh_rsp_t
*gecko_cmd_mesh_test_prov_prepare_key_refresh(aes_key_128   net_key,   uint8   app_keys_len,   const   uint8
*app_keys_data);

/* Response id */
gecko_rsp_mesh_test_prov_prepare_key_refresh_id

/* Response structure */
struct gecko_msg_mesh_test_prov_prepare_key_refresh_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.25 cmd_mesh_test_send_beacons

Send secure network beacons for every network key on the device, regardless of beacon configuration state or how many beacons sent by other devices have been observed.

**Table 2.1107. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x09 | method | Message ID |

**Table 2.1108. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_send_beacons_rsp_t *gecko_cmd_mesh_test_send_beacons();

/* Response id */
gecko_rsp_mesh_test_send_beacons_id

/* Response structure */
struct gecko_msg_mesh_test_send_beacons_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.26 cmd_mesh_test_set_adv_bearer_state

Disable or enable advertisement bearer for sending.

**Table 2.1109. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1f | method | Message ID |
| 4 | uint8 | state | 0: disable advertisement, 1: enable advertisement. |

**Table 2.1110. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_set_adv_bearer_state_rsp_t *gecko_cmd_mesh_test_set_adv_bearer_state(uint8 state);

/* Response id */
gecko_rsp_mesh_test_set_adv_bearer_state_id

/* Response structure */
struct gecko_msg_mesh_test_set_adv_bearer_state_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.27 cmd_mesh_test_set_adv_scan_params

Set non-default advertisement and scanning parameters used in mesh communications. Call this command before node initialization or Provisioner initialization for the settings to take effect.

**Table 2.1111. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | adv_interval_min | Minimum advertisement interval. Value is in units of 0.625 ms. Default value is 1 (0.625 ms). |
| 6-7 | uint16 | adv_interval_max | Maximum advertisement interval. Value is in units of 0.625 ms. Must be equal to or greater than the minimum interval. Default value is 32 (20 ms). |
| 8 | uint8 | adv_repeat_packets | Number of times to repeat each packet on all selected advertisement channels. Range: 1-5. Default value is 1. |
| 9 | uint8 | adv_use_random_address | Bluetooth address type. Range: 0: use public address, 1: use random address. Default value: 0 (public address). |
| 10 | uint8 | adv_channel_map | Advertisement channel selection bitmask. Range: 0x1-0x7. Default value: 7 (all channels) |
| 11-12 | uint16 | scan_interval | Scan interval. Value is in units of 0.625 ms. Range: 0x0004 to 0x4000 (time range of 2.5 ms to 10.24 s). Default value is 160 (100 ms). |
| 13-14 | uint16 | scan_window | Scan window. Value is in units of 0.625 ms. Must be equal to or less than the scan interval. |

**Table 2.1112. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_set_adv_scan_params_rsp_t    *gecko_cmd_mesh_test_set_adv_scan_params(uint16
adv_interval_min,  uint16  adv_interval_max,  uint8  adv_repeat_packets,  uint8  adv_use_random_address,  uint8
adv_channel_map, uint16 scan_interval, uint16 scan_window);

/* Response id */
gecko_rsp_mesh_test_set_adv_scan_params_id
```

```
/* Response structure */
struct gecko_msg_mesh_test_set_adv_scan_params_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.28 cmd_mesh_test_set_element_seqnum

Set current sequence number of an element.

**Table 2.1113. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x27 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-9 | uint32 | seqnum | Sequence number to set on the target element |

**Table 2.1114. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x27 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_set_element_seqnum_rsp_t *gecko_cmd_mesh_test_set_element_seqnum(uint16 elem_index,
uint32 seqnum);

/* Response id */
gecko_rsp_mesh_test_set_element_seqnum_id

/* Response structure */
struct gecko_msg_mesh_test_set_element_seqnum_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.29 cmd_mesh_test_set_iv_index

Set IV index value of the node.

**Table 2.1115. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x26 | method | Message ID |
| 4-7 | uint32 | iv_index | IV Index value to use |

**Table 2.1116. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x26 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_set_iv_index_rsp_t *gecko_cmd_mesh_test_set_iv_index(uint32 iv_index);

/* Response id */
gecko_rsp_mesh_test_set_iv_index_id

/* Response structure */
struct gecko_msg_mesh_test_set_iv_index_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.30 cmd_mesh_test_set_ivupdate_state

Forcefully change the IV update state on the device. Normally, the state changes as a result of an IV index update procedure progressing from one state to the next.

**Table 2.1117. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | state | Whether IV update state should be entered (1) or exited (0). |

**Table 2.1118. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_set_ivupdate_state_rsp_t *gecko_cmd_mesh_test_set_ivupdate_state(uint8 state);

/* Response id */
gecko_rsp_mesh_test_set_ivupdate_state_id

/* Response structure */
struct gecko_msg_mesh_test_set_ivupdate_state_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.31 cmd_mesh_test_set_ivupdate_test_mode

By default, IV index update is limited in how often the update procedure can be performed. This test command can be called to set IV update test mode where any time limits are ignored.

**Table 2.1119. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | mode | Whether test mode is enabled (1) or disabled (0). |

**Table 2.1120. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_test_set_ivupdate_test_mode_rsp_t   *gecko_cmd_mesh_test_set_ivupdate_test_mode(uint8
mode);

/* Response id */
gecko_rsp_mesh_test_set_ivupdate_test_mode_id

/* Response structure */
struct gecko_msg_mesh_test_set_ivupdate_test_mode_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.32 cmd_mesh_test_set_local_config

Set a state to a value in the local Configuration Server model. Use for testing and debugging purposes only.

**Table 2.1121. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | id | The State to modify |
| 6-7 | uint16 | netkey_index | Network key index; ignored for node-wide states |
| 8 | uint8array | value | The new value |

**Table 2.1122. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x18 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_set_local_config_rsp_t  *gecko_cmd_mesh_test_set_local_config(uint16  id,  uint16
netkey_index, uint8 value_len, const uint8 *value_data);

/* Response id */
gecko_rsp_mesh_test_set_local_config_id

/* Response structure */
struct gecko_msg_mesh_test_set_local_config_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.33 cmd_mesh_test_set_local_heartbeat_publication

Set heartbeat publication state of a local node.

**Table 2.1123. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | publication_address | Heartbeat publication address. The address can't be a virtual address. Note that it can be the unassigned address, in which case the heartbeat publishing is disabled. |
| 6 | uint8 | count_log | Heartbeat publication count setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node will send 2^(n-1) heartbeat messages<br>• **0x12 .. 0xfe:** Prohibited<br>• **0xff:** Hearbeat messages are sent indefinitely |
| 7 | uint8 | period_log | Heartbeat publication period setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not sent<br>• **0x01 .. 0x11:** Node will send a heartbeat message every 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |
| 8 | uint8 | ttl | Time-to-live parameter for heartbeat messages |
| 9-10 | uint16 | features | Heartbeat trigger setting. For bits set in the bitmask, reconfiguration of the node feature associated with the bit will result in the node emitting a heartbeat message. Valid values are as follows:<br>• **Bit 0:** Relay feature<br>• **Bit 1:** Proxy feature<br>• **Bit 2:** Friend feature<br>• **Bit 3:** Low power feature<br>Remaining bits are reserved for future use. |
| 11-12 | uint16 | publication_net-key_index | Index of the network key used to encrypt heartbeat messages |

**Table 2.1124. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x17 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_test_set_local_heartbeat_publication_rsp_t
*gecko_cmd_mesh_test_set_local_heartbeat_publication(uint16   publication_address,   uint8   count_log,   uint8
period_log, uint8 ttl, uint16 features, uint16 publication_netkey_index);

/* Response id */
gecko_rsp_mesh_test_set_local_heartbeat_publication_id

/* Response structure */
struct gecko_msg_mesh_test_set_local_heartbeat_publication_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.34 cmd_mesh_test_set_local_heartbeat_subscription

Set local node heartbeat subscription parameters. Normally heartbeat subscription is controlled by the Provisioner.

**Table 2.1125. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | subscription_source | Source address for heartbeat messages. Must be either a unicast address or the unassigned address, in which case heartbeat messages are not processed. |
| 6-7 | uint16 | subscription_destination | Destination address for heartbeat messages. The address must be either the unicast address of the primary element of the node, a group address, or the unassigned address. If it is the unassigned address, heartbeat messages are not processed. |
| 8 | uint8 | period_log | Heartbeat subscription period setting. Valid values are as follows:<br>• **0x00:** Heartbeat messages are not received<br>• **0x01 .. 0x11:** Node will receive heartbeat messages for 2^(n-1) seconds<br>• **0x12 .. 0xff:** Prohibited |

**Table 2.1126. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                      gecko_msg_mesh_test_set_local_heartbeat_subscription_rsp_t
*gecko_cmd_mesh_test_set_local_heartbeat_subscription(uint16        subscription_source,        uint16
subscription_destination, uint8 period_log);

/* Response id */
gecko_rsp_mesh_test_set_local_heartbeat_subscription_id

/* Response structure */
struct gecko_msg_mesh_test_set_local_heartbeat_subscription_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.35 cmd_mesh_test_set_local_model_pub

Set a local model's publication address, key, and parameters.

**Table 2.1127. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | appkey_index | The application key index to use for the application messages published |
| 8-9 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for Bluetooth SIG models. |
| 10-11 | uint16 | model_id | Model ID |
| 12-13 | uint16 | pub_address | The address to publish to |
| 14 | uint8 | ttl | Time-to-Live value for published messages |
| 15 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 16 | uint8 | retrans | Retransmission count and interval, which controls number of times that the model re-publishes the same message after the initial publish transmission and the cadence of retransmissions.<br>Retransmission count is encoded in the three low bits of the value, ranging from 0 to 7. Default value is 0 (no retransmissions).<br>Retransmission interval is encoded in the five high bits of the value, ranging from 0 to 31, in 50-millisecond units. Value of 0 corresponds to 50 ms, while value of 31 corresponds to 1600 ms. |
| 17 | uint8 | credentials | Friendship credentials flag |

**Table 2.1128. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x11 | method | Message ID |

| Byte | Type | Name | Description |
|---|---|---|---|
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_test_set_local_model_pub_rsp_t    *gecko_cmd_mesh_test_set_local_model_pub(uint16
elem_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id, uint16 pub_address, uint8 ttl, uint8
period, uint8 retrans, uint8 credentials);

/* Response id */
gecko_rsp_mesh_test_set_local_model_pub_id

/* Response structure */
struct gecko_msg_mesh_test_set_local_model_pub_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.36 cmd_mesh_test_set_local_model_pub_va

Set a model's publication virtual address, key, and parameters.

**Table 2.1129. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0d | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is the primary element |
| 6-7 | uint16 | appkey_index | The application key index to use for the published messages |
| 8-9 | uint16 | vendor_id | Vendor ID of the configured model. Use 0xffff for Bluetooth SIG models. |
| 10-11 | uint16 | model_id | Model ID of the configured model |
| 12 | uint8 | ttl | Publication time-to-live value |
| 13 | uint8 | period | Publication period encoded as step count and step resolution. The encoding is as follows:<br>• **Bits 0..5:** Step count<br>• **Bits 6..7:** Step resolution:<br>  • 00: 100 milliseconds<br>  • 01: 1 second<br>  • 10: 10 seconds<br>  • 11: 10 minutes |
| 14 | uint8 | retrans | See documentation of local model publication set command for details. |
| 15 | uint8 | credentials | Friendship credentials flag |
| 16 | uint8array | pub_address | The Label UUID to publish to. The byte array must be exactly 16 bytes long. |

**Table 2.1130. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_set_local_model_pub_va_rsp_t  *gecko_cmd_mesh_test_set_local_model_pub_va(uint16
elem_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id, uint8 ttl, uint8 period, uint8 retrans,
```

```
uint8 credentials, uint8 pub_address_len, const uint8 *pub_address_data);

/* Response id */
gecko_rsp_mesh_test_set_local_model_pub_va_id

/* Response structure */
struct gecko_msg_mesh_test_set_local_model_pub_va_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.37 cmd_mesh_test_set_nettx

Set the network transmit state of a node locally. Normally, the network transmit state is controlled by the Provisioner. This command overrides any setting done by the Provisioner.

**Table 2.1131. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | count | Number of network layer transmissions beyond the initial one. Range: 0-7. |
| 5 | uint8 | interval | Transmit interval steps. The interval between transmissions is a random value between 10*(1+steps) and 10*(2+steps) milliseconds. For example, for a value of 2 the interval would be between 30 and 40 milliseconds. Range: 0-31. |

**Table 2.1132. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_set_nettx_rsp_t *gecko_cmd_mesh_test_set_nettx(uint8 count, uint8 interval);

/* Response id */
gecko_rsp_mesh_test_set_nettx_id

/* Response structure */
struct gecko_msg_mesh_test_set_nettx_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.38 cmd_mesh_test_set_relay

Set the relay state and the relay retransmit state of a node locally. Normally, these states are controlled by the Provisioner. This command overrides any settings done by the Provisioner.

**Table 2.1133. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | enabled | Indicates whether the relay functionality is enabled on the node (1) or not (0); value indicating disabled (2) can't be set. |
| 5 | uint8 | count | Number of relay transmissions beyond the initial one. Range: 0-7. |
| 6 | uint8 | interval | Relay retransmit interval steps. The interval between transmissions is 10*(1+steps) milliseconds. Range: 0-31. |

**Table 2.1134. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_test_set_relay_rsp_t *gecko_cmd_mesh_test_set_relay(uint8 enabled, uint8 count, uint8
interval);

/* Response id */
gecko_rsp_mesh_test_set_relay_id

/* Response structure */
struct gecko_msg_mesh_test_set_relay_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.39 cmd_mesh_test_set_replay_protection_list_diagnostics

Enable or disable replay protection list diagnostic events. When enabled, events related to the replay protection list changes are generated.

**Table 2.1135. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x2c | method | Message ID |
| 4 | uint8 | enable | Enable (nonzero) or disable (zero) diagnostic events for replay protection list |

**Table 2.1136. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x2c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                    gecko_msg_mesh_test_set_replay_protection_list_diagnostics_rsp_t
*gecko_cmd_mesh_test_set_replay_protection_list_diagnostics(uint8 enable);

/* Response id */
gecko_rsp_mesh_test_set_replay_protection_list_diagnostics_id

/* Response structure */
struct gecko_msg_mesh_test_set_replay_protection_list_diagnostics_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.40 cmd_mesh_test_set_sar_config

Changes the transport layer segmentation and reassembly configuration values. This command must be issued before initializing the Mesh stack or the changes will not take effect.

**Table 2.1137. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x15 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1d | method | Message ID |
| 4-7 | uint32 | incomplete_timer_ms | Maximum timeout before a transaction expires, regardless of other parameters. Value is in milliseconds. Default = 10000 (10 seconds). |
| 8-11 | uint32 | pending_ack_base_ms | Base time to wait at the receiver before sending a transport layer acknowledgment. Value is in milliseconds. Default = 150. |
| 12-15 | uint32 | pending_ack_mul_ms | TTL multiplier to add to the base acknowledgment timer. Value is in milliseconds. Default = 50. |
| 16-19 | uint32 | wait_for_ack_base_ms | Base time to wait for an acknowledgment at the sender before retransmission. Value is in milliseconds. Default = 200. |
| 20-23 | uint32 | wait_for_ack_mul_ms | TTL multiplier to add to the base retransmission timer. Value is in milliseconds. Default = 50. |
| 24 | uint8 | max_send_rounds | Number of attempts to send fragments of a segmented message, including the initial TX. Default = 3. |

**Table 2.1138. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct        gecko_msg_mesh_test_set_sar_config_rsp_t        *gecko_cmd_mesh_test_set_sar_config(uint32
incomplete_timer_ms,  uint32  pending_ack_base_ms,  uint32  pending_ack_mul_ms,  uint32  wait_for_ack_base_ms,
uint32 wait_for_ack_mul_ms, uint8 max_send_rounds);

/* Response id */
gecko_rsp_mesh_test_set_sar_config_id

/* Response structure */
struct gecko_msg_mesh_test_set_sar_config_rsp_t
{
```

```
  uint16 result;
};
```

#### 2.34.1.41 cmd_mesh_test_set_segment_send_delay

Set delay in milliseconds between sending consecutive segments of a segmented message. The default value is 0. Note that this command needs to be called before node initialization or Provisioner initialization for the settings to take effect.

**Table 2.1139. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | delay | Number of milliseconds to delay each segment after the first |

**Table 2.1140. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_test_set_segment_send_delay_rsp_t   *gecko_cmd_mesh_test_set_segment_send_delay(uint8
delay);

/* Response id */
gecko_rsp_mesh_test_set_segment_send_delay_id

/* Response structure */
struct gecko_msg_mesh_test_set_segment_send_delay_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.42 cmd_mesh_test_unbind_local_model_app

Remove a binding between a model and an application key locally.

**Table 2.1141. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | elem_index | The index of the target element, 0 is primary element |
| 6-7 | uint16 | appkey_index | The Appkey to use for binding |
| 8-9 | uint16 | vendor_id | Vendor ID for vendor-specific models. Use 0xffff for SIG models. |
| 10-11 | uint16 | model_id | Model ID |

**Table 2.1142. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_test_unbind_local_model_app_rsp_t  *gecko_cmd_mesh_test_unbind_local_model_app(uint16
elem_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_test_unbind_local_model_app_id

/* Response structure */
struct gecko_msg_mesh_test_unbind_local_model_app_rsp_t
{
  uint16 result;
};
```

#### 2.34.1.43 cmd_mesh_test_update_local_key

Update network or application key value locally.

Copies the existing network key value to the old value and replaces the current value with the given key data.

Note that the standard way to update keys on Provisioner as well as on nodes is to run the key refresh procedure. This command is for debugging only.

**Table 2.1143. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x13 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1c | method | Message ID |
| 4 | uint8 | key_type | 0 for network key, 1 for application key |
| 5-20 | aes_key_128 | key | Key data |
| 21-22 | uint16 | key_index | Index for the key to update |

**Table 2.1144. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x1c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct   gecko_msg_mesh_test_update_local_key_rsp_t   *gecko_cmd_mesh_test_update_local_key(uint8   key_type,
aes_key_128 key, uint16 key_index);

/* Response id */
gecko_rsp_mesh_test_update_local_key_id

/* Response structure */
struct gecko_msg_mesh_test_update_local_key_rsp_t
{
  uint16 result;
};
```

### 2.34.2 mesh_test events

#### 2.34.2.1 evt_mesh_test_local_heartbeat_subscription_complete

Indicates that the heartbeat subscription period is over

**Table 2.1145. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | count | Number of received heartbeat messages |
| 6 | uint8 | hop_min | Minimum observed hop count in heartbeat messages |
| 7 | uint8 | hop_max | Maximum observed hop count in heartbeat messages |

**C Functions**

```
/* Event id */
gecko_evt_mesh_test_local_heartbeat_subscription_complete_id

/* Event structure */
struct gecko_msg_mesh_test_local_heartbeat_subscription_complete_evt_t
{
  uint16 count;,
  uint8 hop_min;,
  uint8 hop_max;
};
```

#### 2.34.2.2 evt_mesh_test_replay_protection_list_entry_cleared

Indication that a replay protection list entry has been cleared

**Table 2.1146. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | address | Source address for the replay protection list entry |

**C Functions**

```
/* Event id */
gecko_evt_mesh_test_replay_protection_list_entry_cleared_id

/* Event structure */
struct gecko_msg_mesh_test_replay_protection_list_entry_cleared_evt_t
{
  uint16 address;
};
```

#### 2.34.2.3 evt_mesh_test_replay_protection_list_entry_set

Indication that a replay protection list entry has been set

**Table 2.1147. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | address | Source address for the replay protection list entry |
| 6 | uint8 | cancel | Nonzero when replay protection list update relates to a cancelled segmented reception |

**C Functions**

```
/* Event id */
gecko_evt_mesh_test_replay_protection_list_entry_set_id

/* Event structure */
struct gecko_msg_mesh_test_replay_protection_list_entry_set_evt_t
{
  uint16 address;,
  uint8 cancel;
};
```

#### 2.34.2.4 evt_mesh_test_replay_protection_list_full

Indication that replay protection list is full when trying to process a message

**Table 2.1148. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x04 | method | Message ID |

**C Functions**

```
/* Event id */
gecko_evt_mesh_test_replay_protection_list_full_id

/* Event structure */
struct gecko_msg_mesh_test_replay_protection_list_full_evt_t
{
};
```

#### 2.34.2.5 evt_mesh_test_replay_protection_list_saved

Indication that replay protection list has been saved

**Table 2.1149. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x22 | class | Message class: Bluetooth Mesh Test Utilities |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | saved_count | Number of entries saved successfully |
| 8-9 | uint16 | total_count | Number of entries in the list in total |

**C Functions**

```
/* Event id */
gecko_evt_mesh_test_replay_protection_list_saved_id

/* Event structure */
struct gecko_msg_mesh_test_replay_protection_list_saved_evt_t
{
  uint16 result;,
  uint16 saved_count;,
  uint16 total_count;
};
```

### 2.34.3 mesh_test enumerations

#### 2.34.3.1 enum_mesh_test_key_type

Specifies the type of a key in key manipulation commands.

**Table 2.1150. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | mesh_test_key_type_net | Network key |
| 1 | mesh_test_key_type_app | Application key |

## 2.35 Bluetooth Mesh Time Client (mesh_time_client)

This class provides the commands and messages to interface with the Time Client model

### 2.35.1 mesh_time_client commands

#### 2.35.1.1 cmd_mesh_time_client_deinit

Deinitializes the Time Client model

**Table 2.1151. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |

**Table 2.1152. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_client_deinit_rsp_t *gecko_cmd_mesh_time_client_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_client_deinit_id

/* Response structure */
struct gecko_msg_mesh_time_client_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.2 cmd_mesh_time_client_get_tai_utc_delta

Sends a TAI-UTC Delta Get message to Time Server

**Table 2.1153. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |

**Table 2.1154. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_client_get_tai_utc_delta_rsp_t *gecko_cmd_mesh_time_client_get_tai_utc_delta(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_time_client_get_tai_utc_delta_id

/* Response structure */
struct gecko_msg_mesh_time_client_get_tai_utc_delta_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.3 cmd_mesh_time_client_get_time

Sends a Time Get message to Time Server

**Table 2.1155. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |

**Table 2.1156. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_time_client_get_time_rsp_t  *gecko_cmd_mesh_time_client_get_time(uint16  elem_index,
uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_time_client_get_time_id

/* Response structure */
struct gecko_msg_mesh_time_client_get_time_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.4 cmd_mesh_time_client_get_time_role

Sends a Time Role Get message to Time Server, which responds with a Time Role Status message.

**Table 2.1157. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |

**Table 2.1158. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_client_get_time_role_rsp_t    *gecko_cmd_mesh_time_client_get_time_role(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_time_client_get_time_role_id

/* Response structure */
struct gecko_msg_mesh_time_client_get_time_role_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.5 cmd_mesh_time_client_get_time_zone

Sends a Time Zone Get message to Time Server

**Table 2.1159. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |

**Table 2.1160. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_client_get_time_zone_rsp_t    *gecko_cmd_mesh_time_client_get_time_zone(uint16
elem_index, uint16 server_address, uint16 appkey_index);

/* Response id */
gecko_rsp_mesh_time_client_get_time_zone_id

/* Response structure */
struct gecko_msg_mesh_time_client_get_time_zone_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.6 cmd_mesh_time_client_init

Initializes the Time Client model

**Table 2.1161. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |

**Table 2.1162. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_client_init_rsp_t *gecko_cmd_mesh_time_client_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_client_init_id

/* Response structure */
struct gecko_msg_mesh_time_client_init_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.7 cmd_mesh_time_client_set_tai_utc_delta

Sends a TAI-UTC Delta Set message to Time Server, which responds with a TAI-UTC Delta Status message.

**Table 2.1163. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10-13 | int32 | tai_utc_delta_new | Upcoming difference between TAI and UTC is seconds. Range is -255 to 32512. |
| 14-21 | uint64 | tai_of_delta_change | TAI Seconds time of the upcoming TAI-UTC Delta change |

**Table 2.1164. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_client_set_tai_utc_delta_rsp_t *gecko_cmd_mesh_time_client_set_tai_utc_delta(uint16
elem_index, uint16 server_address, uint16 appkey_index, int32 tai_utc_delta_new, uint64 tai_of_delta_change);

/* Response id */
gecko_rsp_mesh_time_client_set_tai_utc_delta_id

/* Response structure */
struct gecko_msg_mesh_time_client_set_tai_utc_delta_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.8 cmd_mesh_time_client_set_time

Sends a Time Set message to Time Server

**Table 2.1165. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10-17 | uint64 | tai_seconds | The current TAI time in seconds since the epoch, 40-bit value |
| 18 | uint8 | subsecond | The sub-second in units of 1/256th second. Range is 0-255. |
| 19 | uint8 | uncertainty | The estimated uncertainty in 10-milliseconds steps. Range is 0-255, representing up to 2.55 seconds. |
| 20 | uint8 | time_authority | 0: No Time Authority, the element does not have a trusted source of time such as GPS or NTP. 1: Time Authority, the element has a trusted source of time or a battery-backed properly initialized RTC. Other values are prohibited. |
| 21-24 | int32 | tai_utc_delta | Current difference between TAI and UTC in seconds. Range is -255 to 32512. |
| 25-26 | int16 | time_zone_offset | The local time zone offset in 15-minute increments. Range is -64 to 191, representing -16 to 47.75 hours. |

**Table 2.1166. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_time_client_set_time_rsp_t  *gecko_cmd_mesh_time_client_set_time(uint16  elem_index,
uint16 server_address, uint16 appkey_index, uint64 tai_seconds, uint8 subsecond, uint8 uncertainty, uint8
time_authority, int32 tai_utc_delta, int16 time_zone_offset);

/* Response id */
gecko_rsp_mesh_time_client_set_time_id

/* Response structure */
```

```
struct gecko_msg_mesh_time_client_set_time_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.9 cmd_mesh_time_client_set_time_role

Sends Time Role Get message to Time Server, which responds with a Time Role Status message.

**Table 2.1167. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10 | uint8 | time_role | The Time Role for the element. |

**Table 2.1168. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_client_set_time_role_rsp_t    *gecko_cmd_mesh_time_client_set_time_role(uint16
elem_index, uint16 server_address, uint16 appkey_index, uint8 time_role);

/* Response id */
gecko_rsp_mesh_time_client_set_time_role_id

/* Response structure */
struct gecko_msg_mesh_time_client_set_time_role_rsp_t
{
  uint16 result;
};
```

#### 2.35.1.10 cmd_mesh_time_client_set_time_zone

Sends a Time Zone Set message to Time Server to set the Time Zone New state

**Table 2.1169. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Destination server model address |
| 8-9 | uint16 | appkey_index | The application key index to use. |
| 10-11 | int16 | time_zone_offset_new | Upcoming local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 12-19 | uint64 | tai_of_zone_change | TAI Seconds time of upcoming Time Zone offset change |

**Table 2.1170. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_client_set_time_zone_rsp_t    *gecko_cmd_mesh_time_client_set_time_zone(uint16
elem_index, uint16 server_address, uint16 appkey_index, int16 time_zone_offset_new, uint64 tai_of_zone_change);

/* Response id */
gecko_rsp_mesh_time_client_set_time_zone_id

/* Response structure */
struct gecko_msg_mesh_time_client_set_time_zone_rsp_t
{
  uint16 result;
};
```

#### 2.35.2 mesh_time_client events

#### 2.35.2.1 evt_mesh_time_client_tai_utc_delta_status

Time Status event

**Table 2.1171. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x10 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Unicast address of the server |
| 8-9 | uint16 | client_address | Client address |
| 10-11 | uint16 | appkey_index | App key index. |
| 12-15 | int32 | tai_utc_delta_current | Current difference between TAI and UTC. Range is -255 to 32512. |
| 16-19 | int32 | tai_utc_delta_new | Upcoming difference between TAI and UTC in seconds. Range is -255 to 32512. |
| 20-27 | uint64 | tai_of_delta_change | TAI seconds time of the upcoming TAI-UTC delta change |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_client_tai_utc_delta_status_id

/* Event structure */
struct gecko_msg_mesh_time_client_tai_utc_delta_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  int32 tai_utc_delta_current;,
  int32 tai_utc_delta_new;,
  uint64 tai_of_delta_change;
};
```

#### 2.35.2.2 evt_mesh_time_client_time_role_status

Time Status event

**Table 2.1172. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Unicast address of the server |
| 8-9 | uint16 | client_address | Client address |
| 10-11 | uint16 | appkey_index | App key index. |
| 12 | uint8 | time_role | The Time Role for the element |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_client_time_role_status_id

/* Event structure */
struct gecko_msg_mesh_time_client_time_role_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint8 time_role;
};
```

#### 2.35.2.3 evt_mesh_time_client_time_status

Time Status event

**Table 2.1173. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Unicast address of the server |
| 8-9 | uint16 | client_address | Client address |
| 10-11 | uint16 | appkey_index | App key index. |
| 12-19 | uint64 | tai_seconds | The current TAI time in seconds |
| 20 | uint8 | subsecond | The sub-second time in units of 1/256th second |
| 21 | uint8 | uncertainty | The estimated uncertainty in 10-millisecond steps |
| 22 | uint8 | time_authority | 0 = No Time Authority, 1 = Time Authority |
| 23-26 | int32 | tai_utc_delta | Current difference between TAI and UTC in seconds. Range is -255 to 32512. |
| 27-28 | int16 | time_zone_offset | The local time zone offset in 15-minute increments. Range is -64 to 191, representing -16 to 47.75 hours. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_client_time_status_id

/* Event structure */
struct gecko_msg_mesh_time_client_time_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  uint64 tai_seconds;,
  uint8 subsecond;,
  uint8 uncertainty;,
  uint8 time_authority;,
  int32 tai_utc_delta;,
  int16 time_zone_offset;
};
```

#### 2.35.2.4 evt_mesh_time_client_time_zone_status

Time Status event

**Table 2.1174. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x53 | class | Message class: Bluetooth Mesh Time Client |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Client model element index |
| 6-7 | uint16 | server_address | Unicast address of the server |
| 8-9 | uint16 | client_address | Client address |
| 10-11 | uint16 | appkey_index | App key index. |
| 12-13 | int16 | time_zone_offset_current | Current local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 14-15 | int16 | time_zone_offset_new | Upcoming local time zone offset |
| 16-23 | uint64 | tai_of_zone_change | TAI seconds time of the upcoming Time Zone offset change |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_client_time_zone_status_id

/* Event structure */
struct gecko_msg_mesh_time_client_time_zone_status_evt_t
{
  uint16 elem_index;,
  uint16 server_address;,
  uint16 client_address;,
  uint16 appkey_index;,
  int16 time_zone_offset_current;,
  int16 time_zone_offset_new;,
  uint64 tai_of_zone_change;
};
```

#### 2.35.3 mesh_time_client enumerations

#### 2.35.3.1 enum_mesh_time_client_time_roles

These values define the Time Role types used by the stack.

**Table 2.1175. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | mesh_time_client_time_role_none | The element does not participate in propagation of time information. |
| 1 | mesh_time_client_time_role_authority | The element publishes Time Status messages but does not process received Time Status messages. |
| 2 | mesh_time_client_time_role_relay | The element processes received and publishes Time Status messages. |
| 3 | mesh_time_client_time_role_client | The element does not publish but processes received Time Status messages. |

## 2.36 Bluetooth Mesh Time Server (mesh_time_server)

This class provides the commands and messages to interface with the Time Server model

### 2.36.1 mesh_time_server commands

#### 2.36.1.1 cmd_mesh_time_server_deinit

Deinitializes the Time Server model

**Table 2.1176. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1177. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_server_deinit_rsp_t *gecko_cmd_mesh_time_server_deinit(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_server_deinit_id

/* Response structure */
struct gecko_msg_mesh_time_server_deinit_rsp_t
{
  uint16 result;
};
```

#### 2.36.1.2 cmd_mesh_time_server_get_datetime

Returns the date and time from the Time Server

**Table 2.1178. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1179. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | year | Year |
| 8 | uint8 | month | Month |
| 9 | uint8 | day | Day |
| 10 | uint8 | hour | Hour |
| 11 | uint8 | min | Minutes |
| 12 | uint8 | sec | Seconds |
| 13-14 | uint16 | msec | Milliseconds |
| 15-16 | int16 | timezone | Local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 17 | uint8 | day_of_week | Day of week, 0..6 represents Monday to Sunday |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_server_get_datetime_rsp_t    *gecko_cmd_mesh_time_server_get_datetime(uint16
elem_index);

/* Response id */
gecko_rsp_mesh_time_server_get_datetime_id

/* Response structure */
struct gecko_msg_mesh_time_server_get_datetime_rsp_t
{
  uint16 result;,
  uint16 year;,
```

```
  uint8 month;,
  uint8 day;,
  uint8 hour;,
  uint8 min;,
  uint8 sec;,
  uint16 msec;,
  int16 timezone;,
  uint8 day_of_week;
};
```

#### 2.36.1.3 cmd_mesh_time_server_get_tai_utc_delta_new

Get the upcoming TAI-UTC delta for the element

**Table 2.1180. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1181. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-9 | int32 | new_delta | Upcoming difference between TAI and UTC in seconds |
| 10-17 | uint64 | tai_of_delta_change | Absolute TAI time when the TAI-UTC Delta will change from Current to New |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_time_server_get_tai_utc_delta_new_rsp_t
*gecko_cmd_mesh_time_server_get_tai_utc_delta_new(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_server_get_tai_utc_delta_new_id

/* Response structure */
struct gecko_msg_mesh_time_server_get_tai_utc_delta_new_rsp_t
{
  uint16 result;,
  int32 new_delta;,
  uint64 tai_of_delta_change;
};
```

#### 2.36.1.4 cmd_mesh_time_server_get_time

Get the current time from Time Server>

**Table 2.1182. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1183. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-13 | uint64 | tai_seconds | The current TAI time in seconds |
| 14 | uint8 | subsecond | The sub-second time in units of 1/256th second |
| 15 | uint8 | uncertainty | The estimated uncertainty in 10 millisecond steps |
| 16 | uint8 | time_authority | 0 = No Time Authority, 1 = Time Authority |
| 17-18 | int16 | time_zone_offset | Current local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 19-22 | int32 | tai_utc_delta | Current difference between TAI and UTC in seconds. Range is -255 to 32512. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_server_get_time_rsp_t *gecko_cmd_mesh_time_server_get_time(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_server_get_time_id

/* Response structure */
struct gecko_msg_mesh_time_server_get_time_rsp_t
{
  uint16 result;,
  uint64 tai_seconds;,
  uint8 subsecond;,
  uint8 uncertainty;,
  uint8 time_authority;,
  int16 time_zone_offset;,
  int32 tai_utc_delta;
};
```

#### 2.36.1.5 cmd_mesh_time_server_get_time_role

Get Time Role for the element

**Table 2.1184. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1185. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | time_role | Time Role of the element |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_server_get_time_role_rsp_t    *gecko_cmd_mesh_time_server_get_time_role(uint16
elem_index);

/* Response id */
gecko_rsp_mesh_time_server_get_time_role_id

/* Response structure */
struct gecko_msg_mesh_time_server_get_time_role_rsp_t
{
  uint16 result;,
  uint8 time_role;
};
```

#### 2.36.1.6 cmd_mesh_time_server_get_time_zone_offset_new

Get the upcoming time zone offset from Time Server

**Table 2.1186. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1187. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | int16 | new_offset | Upcoming local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 8-15 | uint64 | tai_of_zone_change | Absolute TAI time when the Time Zone Offset will change from Current to New |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_time_server_get_time_zone_offset_new_rsp_t
*gecko_cmd_mesh_time_server_get_time_zone_offset_new(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_server_get_time_zone_offset_new_id

/* Response structure */
struct gecko_msg_mesh_time_server_get_time_zone_offset_new_rsp_t
{
  uint16 result;,
  int16 new_offset;,
  uint64 tai_of_zone_change;
};
```

#### 2.36.1.7 cmd_mesh_time_server_init

Initializes the Time Server model

**Table 2.1188. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |

**Table 2.1189. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_time_server_init_rsp_t *gecko_cmd_mesh_time_server_init(uint16 elem_index);

/* Response id */
gecko_rsp_mesh_time_server_init_id

/* Response structure */
struct gecko_msg_mesh_time_server_init_rsp_t
{
  uint16 result;
};
```

#### 2.36.1.8 cmd_mesh_time_server_set_tai_utc_delta_new

Sets the upcoming TAI-UTC delta for the element

**Table 2.1190. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-9 | int32 | new_delta | Upcoming difference between TAI and UTC in seconds |
| 10-17 | uint64 | tai_of_delta_change | Absolute TAI time when the TAI-UTC Delta will change from Current to New |

**Table 2.1191. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_time_server_set_tai_utc_delta_new_rsp_t
*gecko_cmd_mesh_time_server_set_tai_utc_delta_new(uint16     elem_index,     int32     new_delta,     uint64
tai_of_delta_change);

/* Response id */
gecko_rsp_mesh_time_server_set_tai_utc_delta_new_id

/* Response structure */
struct gecko_msg_mesh_time_server_set_tai_utc_delta_new_rsp_t
{
  uint16 result;
};
```

#### 2.36.1.9 cmd_mesh_time_server_set_time

Sets the current time for the element.

**Table 2.1192. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-13 | uint64 | tai_seconds | The current TAI time in seconds |
| 14 | uint8 | subsecond | The sub-second time in units of 1/256th second |
| 15 | uint8 | uncertainty | The estimated uncertainty in 10 millisecond steps |
| 16 | uint8 | time_authority | 0 = No Time Authority, 1 = Time Authority |
| 17-18 | int16 | time_zone_offset | Current local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 19-22 | int32 | tai_utc_delta | Current difference between TAI and UTC in seconds. Range is -255 to 32512. |

**Table 2.1193. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_time_server_set_time_rsp_t  *gecko_cmd_mesh_time_server_set_time(uint16  elem_index,
uint64 tai_seconds, uint8 subsecond, uint8 uncertainty, uint8 time_authority, int16 time_zone_offset, int32
tai_utc_delta);

/* Response id */
gecko_rsp_mesh_time_server_set_time_id

/* Response structure */
struct gecko_msg_mesh_time_server_set_time_rsp_t
{
  uint16 result;
};
```

#### 2.36.1.10 cmd_mesh_time_server_set_time_role

Sets Time Role for the element

**Table 2.1194. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6 | uint8 | time_role | Time Role of the element |

**Table 2.1195. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_mesh_time_server_set_time_role_rsp_t    *gecko_cmd_mesh_time_server_set_time_role(uint16
elem_index, uint8 time_role);

/* Response id */
gecko_rsp_mesh_time_server_set_time_role_id

/* Response structure */
struct gecko_msg_mesh_time_server_set_time_role_rsp_t
{
  uint16 result;
};
```

#### 2.36.1.11 cmd_mesh_time_server_set_time_zone_offset_new

Sets the upcoming time zone offset for the element

**Table 2.1196. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | int16 | new_offset | Upcoming local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 8-15 | uint64 | tai_of_zone_change | Absolute TAI time when the Time Zone Offset will change from Current to New |

**Table 2.1197. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                        gecko_msg_mesh_time_server_set_time_zone_offset_new_rsp_t
*gecko_cmd_mesh_time_server_set_time_zone_offset_new(uint16    elem_index,    int16    new_offset,    uint64
tai_of_zone_change);

/* Response id */
gecko_rsp_mesh_time_server_set_time_zone_offset_new_id

/* Response structure */
struct gecko_msg_mesh_time_server_set_time_zone_offset_new_rsp_t
{
  uint16 result;
};
```

### 2.36.2 mesh_time_server events

#### 2.36.2.1 evt_mesh_time_server_tai_utc_delta_updated

Indicates that the upcoming TAI-UTC Delta has been updated by external source

**Table 2.1198. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-9 | int32 | tai_utc_delta_current | Current difference between TAI and UTC is seconds. Range is -255 to 32512. |
| 10-13 | int32 | tai_utc_delta_new | Upcoming difference between TAI and UTC is seconds. Range is -255 to 32512. |
| 14-21 | uint64 | tai_of_delta_change | TAI Seconds time of the upcoming TAI-UTC Delta change |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_server_tai_utc_delta_updated_id

/* Event structure */
struct gecko_msg_mesh_time_server_tai_utc_delta_updated_evt_t
{
  uint16 elem_index;,
  int32 tai_utc_delta_current;,
  int32 tai_utc_delta_new;,
  uint64 tai_of_delta_change;
};
```

#### 2.36.2.2 evt_mesh_time_server_time_role_updated

**Table 2.1199. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6 | uint8 | time_role | Time Role |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_server_time_role_updated_id

/* Event structure */
struct gecko_msg_mesh_time_server_time_role_updated_evt_t
{
  uint16 elem_index;,
  uint8 time_role;
};
```

#### 2.36.2.3 evt_mesh_time_server_time_updated

Indicates that Time State has been updated by external source, either Time Set message from a Time Client, or a Time Status message

**Table 2.1200. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-13 | uint64 | tai_seconds | The current TAI time in seconds since the epoch, 40-bit value |
| 14 | uint8 | subsecond | The sub-second in units of 1/256th second. Range is 0-255. |
| 15 | uint8 | uncertainty | The estimated uncertainty in 10-milliseconds steps. Range is 0-255, representing up 2.55 seconds. |
| 16 | uint8 | time_authority | 0: No Time Authority, the element does not have a trusted source of time such as GPS or NTP. 1: Time Authority, the element has a trusted source of time or a battery-backed properly initialized RTC. Other values are prohibited. |
| 17-20 | int32 | tai_utc_delta | Current difference between TAI and UTC in seconds. Range is -255 to 32512. |
| 21-22 | int16 | time_zone_offset | The local time zone offset in 15-minute increments. Range is -64 to 191, representing -16 to 47.75 hours. |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_server_time_updated_id

/* Event structure */
struct gecko_msg_mesh_time_server_time_updated_evt_t
{
  uint16 elem_index;,
  uint64 tai_seconds;,
  uint8 subsecond;,
  uint8 uncertainty;,
  uint8 time_authority;,
  int32 tai_utc_delta;,
  int16 time_zone_offset;
};
```

#### 2.36.2.4 evt_mesh_time_server_time_zone_offset_updated

Indicated that upcoming time zone offset change has been updated by external source

**Table 2.1201. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x52 | class | Message class: Bluetooth Mesh Time Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Server model element index |
| 6-7 | int16 | time_zone_offset_current | Current local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 8-9 | int16 | time_zone_offset_new | Upcoming local time zone offset. Range is -64 to 191, representing -16 to 47.75 hours. |
| 10-17 | uint64 | tai_of_zone_change | Absolute TAI time when the Time Zone Offset will change from Current to New |

**C Functions**

```
/* Event id */
gecko_evt_mesh_time_server_time_zone_offset_updated_id

/* Event structure */
struct gecko_msg_mesh_time_server_time_zone_offset_updated_evt_t
{
  uint16 elem_index;,
  int16 time_zone_offset_current;,
  int16 time_zone_offset_new;,
  uint64 tai_of_zone_change;
};
```

## 2.37 Bluetooth Mesh Vendor Model (mesh_vendor_model)

Vendor model API provides functionality to send and receive vendor-specific messages.

Throughout the API, the manipulated model is identified by its element address, vendor ID, and model ID.

The API has functions for sending, receiving, and publishing messages. The application has to implement additional complex functionality (state machines or other model-specific logic).

The stack will handle Mesh transaction layer segmentation and reassembly automatically if the messages sent are long enough to require it.

Note that as the application layer overhead for vendor messages is three bytes (vendor ID and opcode) and the access layer MIC is at least four bytes, the longest vendor application payload which can be sent using an unsegmented transport layer PDU is eight bytes. On the other hand, the longest vendor application payload which can be sent using transport layer segmentation is 377 bytes (fragmented into 32 segments).

### 2.37.1 mesh_vendor_model commands

#### 2.37.1.1 cmd_mesh_vendor_model_clear_publication

Clear vendor model publication message.

Clearing the model publication message disables model publishing, which can be re-enabled by defining the publication message using the set vendor model publication command.

**Table 2.1202. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | elem_index | Publishing model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the model |
| 8-9 | uint16 | model_id | Model ID of the model |

**Table 2.1203. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct                          gecko_msg_mesh_vendor_model_clear_publication_rsp_t
*gecko_cmd_mesh_vendor_model_clear_publication(uint16 elem_index, uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_vendor_model_clear_publication_id

/* Response structure */
struct gecko_msg_mesh_vendor_model_clear_publication_rsp_t
{
  uint16 result;
};
```

#### 2.37.1.2 cmd_mesh_vendor_model_deinit

Deinitialize the model. After this call, the model cannot be used until it is initialized again. See initialization command.

**Table 2.1204. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | elem_index | Model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the model |
| 8-9 | uint16 | model_id | Model ID of the model |

**Table 2.1205. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_vendor_model_deinit_rsp_t *gecko_cmd_mesh_vendor_model_deinit(uint16 elem_index, uint16
vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_vendor_model_deinit_id

/* Response structure */
struct gecko_msg_mesh_vendor_model_deinit_rsp_t
{
  uint16 result;
};
```

### 2.37.1.3 cmd_mesh_vendor_model_init

Initialize the vendor model. This function has to be called before the model can be used. Note that the model can be deinitialized if it is not needed anymore; see deinitialization command.

Opcodes that the model is able to receive at initialization must be defined. This enables the stack to pass only valid messages up to the model during runtime. Per Mesh specification there are up to 64 opcodes per vendor, ranging from 0 to 63. Specifying opcodes outside of that range will result in an error response. Duplicate opcodes in the array do not result in an error, but will of course be recorded only once.

**Table 2.1206. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | elem_index | Model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the model |
| 8-9 | uint16 | model_id | Model ID of the model |
| 10 | uint8 | publish | Indicates if the model is a publish model (non-zero) or not (zero). |
| 11 | uint8array | opcodes | Array of opcodes the model can handle |

**Table 2.1207. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_vendor_model_init_rsp_t  *gecko_cmd_mesh_vendor_model_init(uint16  elem_index,  uint16
vendor_id, uint16 model_id, uint8 publish, uint8 opcodes_len, const uint8 *opcodes_data);

/* Response id */
gecko_rsp_mesh_vendor_model_init_id

/* Response structure */
struct gecko_msg_mesh_vendor_model_init_rsp_t
{
  uint16 result;
};
```

#### 2.37.1.4 cmd_mesh_vendor_model_publish

Publish vendor model publication message.

Sends the stored publication message to the network using the application key and destination address stored in the model publication parameters.

**Table 2.1208. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | elem_index | Publishing model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the model |
| 8-9 | uint16 | model_id | Model ID of the model |

**Table 2.1209. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_vendor_model_publish_rsp_t  *gecko_cmd_mesh_vendor_model_publish(uint16  elem_index,
uint16 vendor_id, uint16 model_id);

/* Response id */
gecko_rsp_mesh_vendor_model_publish_id

/* Response structure */
struct gecko_msg_mesh_vendor_model_publish_rsp_t
{
  uint16 result;
};
```

#### 2.37.1.5 cmd_mesh_vendor_model_send

Send vendor-specific data.

Note that, because of bgapi event length restrictions the message sent may need to be fragmented into several commands. If this is the case, the application must issue the commands in the correct order and mark the command carrying the last message fragment with the final flag set to a non-zero value. The stack will not start sending the message until the complete message is provided by the application. Fragments from multiple messages must not be interleaved.

**Table 2.1210. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0f | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Sending model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the sending model |
| 8-9 | uint16 | model_id | Model ID of the sending model |
| 10-11 | uint16 | destination_address | Destination address of the message. It can be a unicast address, a group address, or a virtual address. |
| 12 | int8 | va_index | Index of the destination Label UUID (used only is the destination address is a virtual address) |
| 13-14 | uint16 | appkey_index | The application key index used |
| 15 | uint8 | nonrelayed | If the message is a response to a received message, set this parameter according to what was received in the receive event. Otherwise, set to non-zero if the message affects only devices in the immediate radio neighborhood. |
| 16 | uint8 | opcode | Message opcode |
| 17 | uint8 | final | Indicates whether this payload chunk is the final one of the message or whether more will follow. |
| 18 | uint8array | payload | Payload data (either complete or partial; see final parameter). |

**Table 2.1211. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_mesh_vendor_model_send_rsp_t  *gecko_cmd_mesh_vendor_model_send(uint16  elem_index,  uint16
vendor_id, uint16 model_id, uint16 destination_address, int8 va_index, uint16 appkey_index, uint8 nonrelayed,
uint8 opcode, uint8 final, uint8 payload_len, const uint8 *payload_data);

/* Response id */
gecko_rsp_mesh_vendor_model_send_id

/* Response structure */
struct gecko_msg_mesh_vendor_model_send_rsp_t
{
  uint16 result;
};
```

#### 2.37.1.6 cmd_mesh_vendor_model_set_publication

Set vendor model publication message.

The model publication message will be sent out when model publication occurs either periodically (if the model is configured for periodic publishing) or explicitly (see vendor model publish command.

Note that, because of bgapi length requirements the message may need to be fragmented over multiple commands. If this is the case, the application must issue the commands in the correct order and mark the command carrying the last message fragment with the final flag set to a non-zero value. The stack will not assign the message to the model until the complete message is provided by the application.

To disable publication the publication message may be erased using the clear vendor model publication message command.

**Table 2.1212. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | elem_index | Publishing model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the model |
| 8-9 | uint16 | model_id | Model ID of the model |
| 10 | uint8 | opcode | Message opcode |
| 11 | uint8 | final | Indicates whether this payload chunk is the final one of the message or whether more will follow. |
| 12 | uint8array | payload | Payload data (either complete or partial; see final parameter). |

**Table 2.1213. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_mesh_vendor_model_set_publication_rsp_t *gecko_cmd_mesh_vendor_model_set_publication(uint16 elem_index, uint16 vendor_id, uint16 model_id, uint8 opcode, uint8 final, uint8 payload_len, const uint8 *payload_data);

/* Response id */
gecko_rsp_mesh_vendor_model_set_publication_id

/* Response structure */
```

```
struct gecko_msg_mesh_vendor_model_set_publication_rsp_t
{
  uint16 result;
};
```

### 2.37.2 mesh_vendor_model events

#### 2.37.2.1 evt_mesh_vendor_model_receive

Vendor model message reception event.

Stack generates this event when a vendor message with a valid opcode is received.

Note that because bgapi event length restrictions, the message may be fragmented into several events. If this is the case, the events will be generated by the stack in the correct order and the last event will be marked with the final flag set to a non-zero value. The application has to concatenate the messages into a single buffer if necessary.

**Table 2.1214. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x11 | lolen | Minimum payload length |
| 2 | 0x19 | class | Message class: Bluetooth Mesh Vendor Model |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | elem_index | Receiving model element index |
| 6-7 | uint16 | vendor_id | Vendor ID of the receiving model |
| 8-9 | uint16 | model_id | Model ID of the receiving model |
| 10-11 | uint16 | source_address | Unicast address of the model which sent the message |
| 12-13 | uint16 | destination_address | Address the message was sent to. It can be either the model element's unicast address or a subscription address of the model |
| 14 | int8 | va_index | Index of the destination Label UUID (valid only is the destination address is a virtual addres) |
| 15-16 | uint16 | appkey_index | The application key index used |
| 17 | uint8 | nonrelayed | If non-zero, indicates that the received message was not relayed (TTL was zero), which means that the devices are within direct radio range of each other. |
| 18 | uint8 | opcode | Message opcode |
| 19 | uint8 | final | Indicates whether this payload chunk is the final one of the message or whether more will follow |
| 20 | uint8array | payload | Payload data (either complete or partial; see final parameter) |

**C Functions**

```
/* Event id */
gecko_evt_mesh_vendor_model_receive_id

/* Event structure */
struct gecko_msg_mesh_vendor_model_receive_evt_t
{
  uint16 elem_index;,
  uint16 vendor_id;,
  uint16 model_id;,
  uint16 source_address;,
  uint16 destination_address;,
  int8 va_index;,
  uint16 appkey_index;,
  uint8 nonrelayed;,
  uint8 opcode;,
  uint8 final;,
  uint8array payload;
};
```

## 2.38 Security Manager (sm)

The commands in this class manage Bluetooth security, including commands for starting and stopping encryption and commands for management of all bonding operations.

The following procedure is used to bond with a remote device:

- Use command sm_configure to configure security requirements and I/O capabilities of this device.
- Use command sm_set_bondable_mode to set this device into bondable mode.
- Use command le_gap_connect to open a connection to the remote device.
- After the connection is open, use command sm_increase_security to encrypt the connection. This will also start the bonding process.

If MITM is required, the application needs to display or ask the user to enter a passkey during the process. See events sm_passkey_display and sm_passkey_request for more information. The following procedure can be used to respond to the bonding initiated by a remote device:

- Use command sm_configure to configure security requirements and I/O capabilities of this device.
- Use command sm_set_bondable_mode to set this device into bondable mode.
- Use command le_gap_start_advertising to set this device into advertising and connectable mode.
- Open a connection to this device from the remote device.
- After the connection is open, start the bonding process on the remote device.

If MITM is required, the application needs to display or ask the user to enter a passkey during the process. See events sm_passkey_display and sm_passkey_request for more information.

### 2.38.1 sm commands

#### 2.38.1.1 cmd_sm_add_to_whitelist

Add device to whitelist, which can be enabled with le_gap_enable_whitelisting.

**Table 2.1215. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x13 | method | Message ID |
| 4-9 | bd_addr | address | Address of the device added to whitelist |
| 10 | uint8 | address_type | Address type of the device added to whitelist |

**Table 2.1216. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_add_to_whitelist_rsp_t *gecko_cmd_sm_add_to_whitelist(bd_addr address, uint8 address_type);

/* Response id */
gecko_rsp_sm_add_to_whitelist_id

/* Response structure */
struct gecko_msg_sm_add_to_whitelist_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.2 cmd_sm_bonding_confirm

Accept or reject the bonding request.

**Table 2.1217. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0e | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | confirm | Acceptance. Values:<br>• **0:** Reject<br>• **1:** Accept bonding request |

**Table 2.1218. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_bonding_confirm_rsp_t *gecko_cmd_sm_bonding_confirm(uint8 connection, uint8 confirm);

/* Response id */
gecko_rsp_sm_bonding_confirm_id

/* Response structure */
struct gecko_msg_sm_bonding_confirm_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.3 cmd_sm_configure

Configure security requirements and I/O capabilities of the system.

**Table 2.1219. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | flags | Security requirement bitmask.<br>Bit 0:<br>• **0:** Allow bonding without MITM protection<br>• **1:** Bonding requires MITM protection<br>Bit 1:<br>• **0:** Allow encryption without bonding<br>• **1:** Encryption requires bonding. Note that this setting will also enable bonding.<br>Bit 2:<br>• **0:** Allow bonding with legacy pairing<br>• **1:** Secure connections only<br>Bit 3:<br>• **0:** Bonding request does not need to be confirmed<br>• **1:** Bonding requests need to be confirmed. Received bonding requests are notified by sm_confirm_bonding events.<br>Bit 4:<br>• **0:** Allow all connections<br>• **1:** Allow connections only from bonded devices<br>Bit 5 to 7: Reserved<br>Default value: 0x00 |
| 5 | uint8 | io_capabilities | I/O Capabilities. See link. |

**Table 2.1220. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_configure_rsp_t *gecko_cmd_sm_configure(uint8 flags, uint8 io_capabilities);

/* Response id */
gecko_rsp_sm_configure_id

/* Response structure */
struct gecko_msg_sm_configure_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.4 cmd_sm_delete_bonding

Delete specified bonding information or whitelist from the persistent store.

**Table 2.1221. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | bonding | Bonding handle |

**Table 2.1222. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_delete_bonding_rsp_t *gecko_cmd_sm_delete_bonding(uint8 bonding);

/* Response id */
gecko_rsp_sm_delete_bonding_id

/* Response structure */
struct gecko_msg_sm_delete_bonding_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.5 cmd_sm_delete_bondings

Delete all bonding information and whitelist from the persistent store.

**Table 2.1223. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x07 | method | Message ID |

**Table 2.1224. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_delete_bondings_rsp_t *gecko_cmd_sm_delete_bondings();

/* Response id */
gecko_rsp_sm_delete_bondings_id

/* Response structure */
struct gecko_msg_sm_delete_bondings_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.6 cmd_sm_enter_passkey

Enter a passkey after receiving a passkey request event.

**Table 2.1225. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | int32 | passkey | Passkey. Valid range: 0-999999. Set -1 to cancel pairing. |

**Table 2.1226. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_enter_passkey_rsp_t *gecko_cmd_sm_enter_passkey(uint8 connection, int32 passkey);

/* Response id */
gecko_rsp_sm_enter_passkey_id

/* Response structure */
struct gecko_msg_sm_enter_passkey_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.7 cmd_sm_increase_security

Enhance the security of a connection to current security requirements. On an unencrypted connection, it will encrypt the connection and will also perform bonding if requested by both devices. On an encrypted connection, it will cause the connection to be re-encrypted.

**Table 2.1227. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**Table 2.1228. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_increase_security_rsp_t *gecko_cmd_sm_increase_security(uint8 connection);

/* Response id */
gecko_rsp_sm_increase_security_id

/* Response structure */
struct gecko_msg_sm_increase_security_rsp_t
{
  uint16 result;
};
```

**Table 2.1229. Events Generated**

| Event | Description |
|---|---|
| le_connection_parameters | Triggered after increasing security has been completed successfully and indicates the latest security mode of the connection. |
| sm_bonded | Triggered if pairing or bonding was performed in this operation and the result is successful. |
| sm_bonding_failed | Triggered if pairing or bonding was performed in this operation and the result has failed. |

#### 2.38.1.8 cmd_sm_list_all_bondings

List all bondings stored in the bonding database. Bondings are reported by the sm_list_bonding_entry event for each bonding and the report is ended with sm_list_all_bondings_complete event. Use only for debugging purposes because reading from the persistent store is relatively slow.

**Table 2.1230. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0b | method | Message ID |

**Table 2.1231. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_list_all_bondings_rsp_t *gecko_cmd_sm_list_all_bondings();

/* Response id */
gecko_rsp_sm_list_all_bondings_id

/* Response structure */
struct gecko_msg_sm_list_all_bondings_rsp_t
{
  uint16 result;
};
```

**Table 2.1232. Events Generated**

| Event | Description |
|---|---|
| sm_list_bonding_entry | Triggered by the command sm_list_all_bondings if bondings exist in the local database. |
| sm_list_all_bondings_complete | Triggered by the sm_list_all_bondings and follows sm_list_bonding_entry events. |

#### 2.38.1.9 cmd_sm_passkey_confirm

Accept or reject the reported passkey confirm value.

**Table 2.1233. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | confirm | Acceptance. Values:<br>• **0:** Reject<br>• **1:** Accept confirm value |

**Table 2.1234. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_passkey_confirm_rsp_t *gecko_cmd_sm_passkey_confirm(uint8 connection, uint8 confirm);

/* Response id */
gecko_rsp_sm_passkey_confirm_id

/* Response structure */
struct gecko_msg_sm_passkey_confirm_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.10 cmd_sm_set_bondable_mode

Set whether the device should accept new bondings. By default, the device does not accept new bondings.

**Table 2.1235. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | bondable | Bondable mode. Values:<br>• **0:** New bondings not accepted<br>• **1:** Bondings allowed<br>Default value: 0 |

**Table 2.1236. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_set_bondable_mode_rsp_t *gecko_cmd_sm_set_bondable_mode(uint8 bondable);

/* Response id */
gecko_rsp_sm_set_bondable_mode_id

/* Response structure */
struct gecko_msg_sm_set_bondable_mode_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.11 cmd_sm_set_debug_mode

Set Security Manager in debug mode. In this mode, the secure connections bonding uses known debug keys, so that the encrypted packet can be opened by Bluetooth protocol analyzer. To disable the debug mode, restart the device.

Bondings made in debug mode are unsecure.

**Table 2.1237. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0f | method | Message ID |

**Table 2.1238. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_set_debug_mode_rsp_t *gecko_cmd_sm_set_debug_mode();

/* Response id */
gecko_rsp_sm_set_debug_mode_id

/* Response structure */
struct gecko_msg_sm_set_debug_mode_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.12 cmd_sm_set_minimum_key_size

Set the minimum allowed key size used for bonding. The default value is 16 bytes.

**Table 2.1239. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x14 | method | Message ID |
| 4 | uint8 | minimum_key_size | Minimum allowed key size for bonding. Range: 7 to 16 |

**Table 2.1240. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_set_minimum_key_size_rsp_t *gecko_cmd_sm_set_minimum_key_size(uint8 minimum_key_size);

/* Response id */
gecko_rsp_sm_set_minimum_key_size_id

/* Response structure */
struct gecko_msg_sm_set_minimum_key_size_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.13 cmd_sm_set_oob_data

Set OOB data (out-of-band encryption data) for legacy pairing for a device. OOB data may be, for example, a PIN code exchanged over an alternate path, such as NFC. The device will not allow any other bonding if OOB data is set. OOB data can't be set simultaneously with secure connections OOB data.

**Table 2.1241. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0a | method | Message ID |
| 4 | uint8array | oob_data | OOB data. To set OOB data, send a 16-byte array. To clear OOB data, send a zero-length array. |

**Table 2.1242. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_sm_set_oob_data_rsp_t  *gecko_cmd_sm_set_oob_data(uint8  oob_data_len,  const  uint8
*oob_data_data);

/* Response id */
gecko_rsp_sm_set_oob_data_id

/* Response structure */
struct gecko_msg_sm_set_oob_data_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.14 cmd_sm_set_passkey

Enter a fixed passkey, which will be used in the sm_passkey_display event.

**Table 2.1243. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x10 | method | Message ID |
| 4-7 | int32 | passkey | Passkey. Valid range: 0-999999. Set -1 to disable and start using random passkeys. |

**Table 2.1244. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_set_passkey_rsp_t *gecko_cmd_sm_set_passkey(int32 passkey);

/* Response id */
gecko_rsp_sm_set_passkey_id

/* Response structure */
struct gecko_msg_sm_set_passkey_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.15 cmd_sm_set_sc_remote_oob_data

Set OOB data and confirm values (out-of-band encryption) received from the remote device for secure connections pairing. OOB data must be enabled with sm_use_sc_oob before setting the remote device OOB data.

**Table 2.1245. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x12 | method | Message ID |
| 4 | uint8array | oob_data | Remote device OOB data and confirm values. To set OOB data, send a 32-byte array. First 16-bytes is OOB data and last 16-bytes the confirm value. To clear OOB data, send a zero-length array. |

**Table 2.1246. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_sm_set_sc_remote_oob_data_rsp_t  *gecko_cmd_sm_set_sc_remote_oob_data(uint8  oob_data_len,
const uint8 *oob_data_data);

/* Response id */
gecko_rsp_sm_set_sc_remote_oob_data_id

/* Response structure */
struct gecko_msg_sm_set_sc_remote_oob_data_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.16 cmd_sm_store_bonding_configuration

Set the maximum allowed bonding count and bonding policy. The maximum number of bondings that can be supported depends on how much user data is stored in the NVM and the NVM size. When bond policy value 1 or 2 is selected the stack will automatically write the new bond, as per the policy, only if the maximum allowed bonding count has been reached. If the stack is not able to write a new bond for any other reason (e.g. nvm full) then an error will be thrown through the bonding_failed event indicating why the bonding could not be written. It is left up to the application to manually release space from the nvm (e.g. by deleting one of the existing bonds or application data) so that a new bond can be saved. The default value is 13.

**Table 2.1247. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | max_bonding_count | Maximum allowed bonding count. Range: 1 to 32 |
| 5 | uint8 | policy_flags | Bonding policy. Values:<br>• **0:** If database is full, new bonding attempts will fail<br>• **1:** New bonding will overwrite the oldest existing bonding<br>• **2:** New bonding will overwrite the existing bonding that was used the longest time ago<br>Default: 0 |

**Table 2.1248. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct    gecko_msg_sm_store_bonding_configuration_rsp_t    *gecko_cmd_sm_store_bonding_configuration(uint8
max_bonding_count, uint8 policy_flags);

/* Response id */
gecko_rsp_sm_store_bonding_configuration_id

/* Response structure */
struct gecko_msg_sm_store_bonding_configuration_rsp_t
{
  uint16 result;
};
```

#### 2.38.1.17 cmd_sm_use_sc_oob

Enable the use of OOB data (out-of-band encryption data) for a device for secure connections pairing. Enabling will generate new OOB data and confirm values, which can be sent to the remote device. After enabling the secure connections OOB data, the remote devices OOB data can be set with sm_set_sc_remote_oob_data. Calling this function will erase any set remote device OOB data and confirm values. The device will not allow any other bonding if OOB data is set. The secure connections OOB data cannot be enabled simultaneously with legacy pairing OOB data.

**Table 2.1249. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x11 | method | Message ID |
| 4 | uint8 | enable | Enable OOB with secure connections pairing. Values:<br>• **0:** disable<br>• **1:** enable |

**Table 2.1250. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x11 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | oob_data | OOB data. 32-byte array. The first 16-bytes contain randomly-generated OOB data and the last 16-bytes confirm value. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sm_use_sc_oob_rsp_t *gecko_cmd_sm_use_sc_oob(uint8 enable);

/* Response id */
gecko_rsp_sm_use_sc_oob_id

/* Response structure */
struct gecko_msg_sm_use_sc_oob_rsp_t
{
  uint16 result;,
  uint8array oob_data;
};
```

### 2.38.2 sm events

#### 2.38.2.1 evt_sm_bonded

Triggered after the pairing or bonding procedure is successfully completed.

**Table 2.1251. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8 | bonding | Bonding handle. Values:<br>• **0xff:** Pairing completed without bonding - the pairing key will be discarded after disconnection.<br>• **Other:** Procedure completed, pairing key stored with given bonding handle |

**C Functions**

```
/* Event id */
gecko_evt_sm_bonded_id

/* Event structure */
struct gecko_msg_sm_bonded_evt_t
{
  uint8 connection;,
  uint8 bonding;
};
```

#### 2.38.2.2 evt_sm_bonding_failed

This event is triggered if the pairing or bonding procedure fails.

**Table 2.1252. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | reason | Describes error that occurred |

**C Functions**

```
/* Event id */
gecko_evt_sm_bonding_failed_id

/* Event structure */
struct gecko_msg_sm_bonding_failed_evt_t
{
  uint8 connection;,
  uint16 reason;
};
```

#### 2.38.2.3 evt_sm_confirm_bonding

Indicates a user request to display that the new bonding request is received and for the user to confirm the request. Use the command sm_bonding_confirm to accept or reject the bonding request.

**Table 2.1253. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | int8 | bonding_handle | Bonding handle for the request. Range: -1 to 31.<br>• NOTE! When the bonding handle is anything other than -1, a bonding already exists for this connection. Overwriting the existing bonding is a potential security risk. |

**C Functions**

```
/* Event id */
gecko_evt_sm_confirm_bonding_id

/* Event structure */
struct gecko_msg_sm_confirm_bonding_evt_t
{
  uint8 connection;,
  int8 bonding_handle;
};
```

#### 2.38.2.4 evt_sm_confirm_passkey

Indicates a request for passkey display and confirmation by the user. Use the command sm_passkey_confirm to accept or reject the displayed passkey.

**Table 2.1254. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | passkey | Passkey. Range: 0 to 999999.<br>• NOTE! When displaying the passkey to the user, prefix the number with zeros to obtain a 6 digit number<br>• Example: Passkey value is 42<br>• Number to display to the user is 000042 |

**C Functions**

```
/* Event id */
gecko_evt_sm_confirm_passkey_id

/* Event structure */
struct gecko_msg_sm_confirm_passkey_evt_t
{
  uint8 connection;,
  uint32 passkey;
};
```

#### 2.38.2.5 evt_sm_list_all_bondings_complete

Triggered by the sm_list_all_bondings and follows sm_list_bonding_entry events.

**Table 2.1255. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x06 | method | Message ID |

**C Functions**

```
/* Event id */
gecko_evt_sm_list_all_bondings_complete_id

/* Event structure */
struct gecko_msg_sm_list_all_bondings_complete_evt_t
{
};
```

#### 2.38.2.6 evt_sm_list_bonding_entry

Triggered by the command sm_list_all_bondings if bondings exist in the local database.

**Table 2.1256. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | bonding | Bonding handle |
| 5-10 | bd_addr | address | Bluetooth address of the remote device |
| 11 | uint8 | address_type | Address type |

**C Functions**

```
/* Event id */
gecko_evt_sm_list_bonding_entry_id

/* Event structure */
struct gecko_msg_sm_list_bonding_entry_evt_t
{
  uint8 bonding;,
  bd_addr address;,
  uint8 address_type;
};
```

#### 2.38.2.7 evt_sm_passkey_display

Indicates a request to display the passkey to the user.

**Table 2.1257. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | passkey | Passkey. Range: 0 to 999999.<br>• NOTE! When displaying the passkey to the user, prefix the number with zeros to obtain a 6 digit number<br>• Example: Passkey value is 42<br>• Number to display to the user is 000042 |

**C Functions**

```
/* Event id */
gecko_evt_sm_passkey_display_id

/* Event structure */
struct gecko_msg_sm_passkey_display_evt_t
{
  uint8 connection;,
  uint32 passkey;
};
```

#### 2.38.2.8 evt_sm_passkey_request

Indicates a request for the passkey prompt displayed on the remote device. Use the command sm_enter_passkey to input the passkey value.

**Table 2.1258. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

**C Functions**

```
/* Event id */
gecko_evt_sm_passkey_request_id

/* Event structure */
struct gecko_msg_sm_passkey_request_evt_t
{
  uint8 connection;
};
```

### 2.38.3 sm enumerations

#### 2.38.3.1 enum_sm_bonding_key

These values define the bonding information of the bonded device stored in the persistent store.

**Table 2.1259. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | sm_bonding_key_ltk | LTK saved in master |
| 2 | sm_bonding_key_addr_public | Public Address |
| 4 | sm_bonding_key_addr_static | Static Address |
| 8 | sm_bonding_key_irk | Identity resolving key for resolvable private addresses |
| 16 | sm_bonding_key_edivrand | EDIV+RAND received from slave |
| 32 | sm_bonding_key_csrk | Connection signature resolving key |
| 64 | sm_bonding_key_masterid | EDIV+RAND sent to master |

#### 2.38.3.2 enum_sm_io_capability

These values define the security management related I/O capabilities supported by the device.

**Table 2.1260. Enumerations**

| Value | Name | Description |
|---|---|---|
| 0 | sm_io_capability_displayonly | Display Only |
| 1 | sm_io_capability_displayyesno | Display with Yes/No-buttons |
| 2 | sm_io_capability_keyboardonly | Keyboard Only |
| 3 | sm_io_capability_noinputnooutput | No Input and No Output |
| 4 | sm_io_capability_keyboarddisplay | Display with Keyboard |

## 2.39 Periodic Advertising Synchronization (sync)

Provides periodic advertising synchronization feature.

When this feature is used, enable event le_gap_extended_scan_response which contains useful information for establishing a synchronization.

### 2.39.1 sync commands

#### 2.39.1.1 cmd_sync_close

Closes a periodic advertising synchronization or cancels an ongoing attempt of establishing a synchronization.

**Table 2.1261. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | sync | Periodic advertising synchronization handle |

**Table 2.1262. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sync_close_rsp_t *gecko_cmd_sync_close(uint8 sync);

/* Response id */
gecko_rsp_sync_close_id

/* Response structure */
struct gecko_msg_sync_close_rsp_t
{
  uint16 result;
};
```

**Table 2.1263. Events Generated**

| Event | Description |
|---|---|
| sync_closed | Triggered after a periodic advertising synchronization has been closed or canceled. |

#### 2.39.1.2 cmd_sync_open

Establish a synchronization with a periodic advertising from the specified advertiser and begin receiving periodic advertising packets. Note that synchronization establishment can only occur when scanning is enabled. While scanning is disabled, no attempt to synchronize will occur.

The application should determine skip and timeout values based on the periodic advertising interval provided by the advertiser. If skip and timeout are used, select appropriate values so that they allow a few receiving attempts. Periodic advertising intervals are reported in event le_gap_extended_scan_response.

**Table 2.1264. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0c | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | adv_sid | Advertising set identifier |
| 5-6 | uint16 | skip | The maximum number of periodic advertising packets that can be skipped after a successful receive. Range: 0x0000 to 0x01F3 |
| 7-8 | uint16 | timeout | The maximum permitted time between successful receives. If this time is exceeded, synchronization is lost. Unit: 10 ms.<br>• Range: 0x06 to 0xFFFF<br>• Unit: 10 ms<br>• Time range: 100 ms ms to 163.84 s |
| 9-14 | bd_addr | address | Address of the advertiser |
| 15 | uint8 | address_type | Advertiser address type. Values:<br>• **0:** Public address<br>• **1:** Random address |

**Table 2.1265. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | sync | A handle that will be assigned to the periodic advertising synchronization after the synchronization is established. This handle is valid only if the result code of this response is 0 (zero). |

**BGLIB C API**

```
/* Function */
struct gecko_msg_sync_open_rsp_t *gecko_cmd_sync_open(uint8 adv_sid, uint16 skip, uint16 timeout, bd_addr
```

```
address, uint8 address_type);

/* Response id */
gecko_rsp_sync_open_id

/* Response structure */
struct gecko_msg_sync_open_rsp_t
{
  uint16 result;,
  uint8 sync;
};
```

**Table 2.1266. Events Generated**

| Event | Description |
|---|---|
| sync_opened | Triggered after the synchronization is established. |
| sync_data | Indicates that a periodic advertisement packet is received. |

### 2.39.2 sync events

#### 2.39.2.1 evt_sync_closed

Indicates that a periodic advertising synchronization lost or a synchronization establishment procedure was canceled.

**Table 2.1267. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | reason | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8 | sync | Periodic advertising synchronization handle |

**C Functions**

```
/* Event id */
gecko_evt_sync_closed_id

/* Event structure */
struct gecko_msg_sync_closed_evt_t
{
  uint16 reason;,
  uint8 sync;
};
```

#### 2.39.2.2 evt_sync_data

Reports a received periodic advertisement packet.

**Table 2.1268. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | sync | Periodic advertising synchronization handle |
| 5 | int8 | tx_power | TX power value in the received packet header. Units: dBm<br>• Valid value range: -127 to 126<br>• Value 127: information unavailable |
| 6 | int8 | rssi | Signal strength indicator (RSSI) in the latest received packet. Units: dBm<br>• Range: -127 to +20 |
| 7 | uint8 | data_status | Data completeness:<br>• **0:** Complete<br>• **1:** Incomplete, more data to come in new events<br>• **2:** Incomplete, data truncated, no more to come |
| 8 | uint8array | data | Periodic advertising data |

**C Functions**

```
/* Event id */
gecko_evt_sync_data_id

/* Event structure */
struct gecko_msg_sync_data_evt_t
{
  uint8 sync;,
  int8 tx_power;,
  int8 rssi;,
  uint8 data_status;,
  uint8array data;
};
```

#### 2.39.2.3 evt_sync_opened

Indicates that a periodic advertising synchronization has been opened.

**Table 2.1269. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x42 | class | Message class: Periodic Advertising Synchronization |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | sync | Periodic advertising synchronization handle |
| 5 | uint8 | adv_sid | Advertising set identifier |
| 6-11 | bd_addr | address | Address of the advertiser |
| 12 | uint8 | address_type | Advertiser address type. Values:<br>• **0:** Public address<br>• **1:** Random address |
| 13 | uint8 | adv_phy | The advertiser PHY. Value:<br>• **1:** 1M PHY<br>• **2:** 2M PHY<br>• **4:** Coded PHY |
| 14-15 | uint16 | adv_interval | The periodic advertising interval. Value in units of 1.25 ms<br>• Range: 0x06 to 0xFFFF<br>• Time range: 7.5 ms to 81.92 s |
| 16-17 | uint16 | clock_accuracy | The advertiser clock accuracy. |

**C Functions**

```
/* Event id */
gecko_evt_sync_opened_id

/* Event structure */
struct gecko_msg_sync_opened_evt_t
{
  uint8 sync;,
  uint8 adv_sid;,
  bd_addr address;,
  uint8 address_type;,
  uint8 adv_phy;,
  uint16 adv_interval;,
  uint16 clock_accuracy;
};
```

### 2.39.3 sync enumerations

#### 2.39.3.1 enum_sync_advertiser_clock_accuracy

These values indicate the advertiser clock accuracy in a periodic advertising synchronization.

**Table 2.1270. Enumerations**

| Value | Name | Description |
|---|---|---|
| 500 | sync_clock_accuracy_500 | Clock accuracy 500 ppm |
| 250 | sync_clock_accuracy_250 | Clock accuracy 250 ppm |
| 150 | sync_clock_accuracy_150 | Clock accuracy 150 ppm |
| 100 | sync_clock_accuracy_100 | Clock accuracy 100 ppm |
| 75 | sync_clock_accuracy_75 | Clock accuracy 75 ppm |
| 50 | sync_clock_accuracy_50 | Clock accuracy 50 ppm |
| 30 | sync_clock_accuracy_30 | Clock accuracy 30 ppm |
| 20 | sync_clock_accuracy_20 | Clock accuracy 20 ppm |

## 2.40 System (system)

Commands and events in this class can be used to access and query the local device.

### 2.40.1 system commands

#### 2.40.1.1 cmd_system_data_buffer_clear

Remove all data from the system data buffer.

**Table 2.1271. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x14 | method | Message ID |

**Table 2.1272. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x14 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_data_buffer_clear_rsp_t *gecko_cmd_system_data_buffer_clear();

/* Response id */
gecko_rsp_system_data_buffer_clear_id

/* Response structure */
struct gecko_msg_system_data_buffer_clear_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.2 cmd_system_data_buffer_write

Write data into the system data buffer. Data will be appended to the end of existing data.

**Table 2.1273. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x12 | method | Message ID |
| 4 | uint8array | data | Data to write |

**Table 2.1274. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x12 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_system_data_buffer_write_rsp_t  *gecko_cmd_system_data_buffer_write(uint8  data_len,  const
uint8 *data_data);

/* Response id */
gecko_rsp_system_data_buffer_write_id

/* Response structure */
struct gecko_msg_system_data_buffer_write_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.3 cmd_system_get_bt_address

Read the Bluetooth public address used by the device.

**Table 2.1275. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x03 | method | Message ID |

**Table 2.1276. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x03 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth public address in little endian format |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_get_bt_address_rsp_t *gecko_cmd_system_get_bt_address();

/* Response id */
gecko_rsp_system_get_bt_address_id

/* Response structure */
struct gecko_msg_system_get_bt_address_rsp_t
{
  bd_addr address;
};
```

#### 2.40.1.4 cmd_system_get_counters

Get packet and error counters. Passing a non-zero value also resets counters.

**Table 2.1277. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0f | method | Message ID |
| 4 | uint8 | reset | Reset counters if the parameter value is not zero. |

**Table 2.1278. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0a | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0f | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6-7 | uint16 | tx_packets | The number of successfully transmitted packets |
| 8-9 | uint16 | rx_packets | The number of successfully received packets |
| 10-11 | uint16 | crc_errors | The number of received packets with CRC errors |
| 12-13 | uint16 | failures | The number of radio failures, such as aborted TX/RX packets, scheduling failures, and so on. |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_get_counters_rsp_t *gecko_cmd_system_get_counters(uint8 reset);

/* Response id */
gecko_rsp_system_get_counters_id

/* Response structure */
struct gecko_msg_system_get_counters_rsp_t
{
  uint16 result;,
  uint16 tx_packets;,
  uint16 rx_packets;,
  uint16 crc_errors;,
  uint16 failures;
};
```

#### 2.40.1.5 cmd_system_get_random_data

Get random data up to 16 bytes.

**Table 2.1279. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0b | method | Message ID |
| 4 | uint8 | length | Length of random data. Maximum length is 16 bytes. |

**Table 2.1280. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | data | Random data |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_get_random_data_rsp_t *gecko_cmd_system_get_random_data(uint8 length);

/* Response id */
gecko_rsp_system_get_random_data_id

/* Response structure */
struct gecko_msg_system_get_random_data_rsp_t
{
  uint16 result;,
  uint8array data;
};
```

#### 2.40.1.6 cmd_system_halt

Force radio to idle state and allow device to sleep. Advertising, scanning, connections, and software timers are halted by this command. Halted operations resume after calling this command with parameter 0. Connections stay alive if system is resumed before connection supervision timeout.

Use this command only for a short time period (a few seconds at maximum). Although it halts Bluetooth activity, all tasks and operations still exist inside the 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:** The software timer is also halted. Hardware interrupts are the only way to wake up from energy mode 2 when the system is halted.

**Table 2.1281. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0c | method | Message ID |
| 4 | uint8 | halt | Values:<br>• **1:** halt<br>• **0:** resume |

**Table 2.1282. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0c | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_halt_rsp_t *gecko_cmd_system_halt(uint8 halt);

/* Response id */
gecko_rsp_system_halt_id

/* Response structure */
struct gecko_msg_system_halt_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.7 cmd_system_hello

Verify whether the communication between the host and the device is functional.

**Table 2.1283. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x00 | method | Message ID |

**Table 2.1284. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_hello_rsp_t *gecko_cmd_system_hello();

/* Response id */
gecko_rsp_system_hello_id

/* Response structure */
struct gecko_msg_system_hello_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.8 cmd_system_linklayer_configure

Send configuration data to the link layer. This command fine tunes low-level Bluetooth operations.

**Table 2.1285. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0e | method | Message ID |
| 4 | uint8 | key | Key to configure |
| 5 | uint8array | data | Configuration data. Length and contents of the data field depend on the key value used. |

**Table 2.1286. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0e | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_system_linklayer_configure_rsp_t  *gecko_cmd_system_linklayer_configure(uint8  key,  uint8
data_len, const uint8 *data_data);

/* Response id */
gecko_rsp_system_linklayer_configure_id

/* Response structure */
struct gecko_msg_system_linklayer_configure_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.9 cmd_system_reset

Reset the system. The command does not have a response but it triggers one of the boot events (normal reset or boot to DFU mode) depending on the selected boot mode.

**Table 2.1287. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | dfu | Boot mode:<br>• **0:** Normal reset<br>• **1:** Boot to UART DFU mode<br>• **2:** Boot to OTA DFU mode |

**BGLIB C API**

```
/* Function */
void *gecko_cmd_system_reset(uint8 dfu);

/* Command does not have a response */
```

**Table 2.1288. Events Generated**

| Event | Description |
|---|---|
| system_boot | Sent after the device has booted in normal mode. |
| dfu_boot | Sent after the device has booted in UART DFU mode. |

#### 2.40.1.10 (deprecated) cmd_system_set_bt_address

**Deprecated** and replaced by system_set_identity_address command.

Set the Bluetooth public address used by the device. A valid address set with this command overrides the default Bluetooth public address programmed at production and is 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. As a result, passing one of them into this command will cause the stack to use the default address in the next system reboot.

**Table 2.1289. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x04 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth public address in little endian format |

**Table 2.1290. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_set_bt_address_rsp_t *gecko_cmd_system_set_bt_address(bd_addr address);

/* Response id */
gecko_rsp_system_set_bt_address_id

/* Response structure */
struct gecko_msg_system_set_bt_address_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.11 cmd_system_set_device_name

Set the device name which will be used during the OTA update. The name will be stored in the persistent store. If the OTA device name is also set in the stack configuration, the name stored in the persistent store is overwritten by the name in the stack configuration during the device boot.

**Table 2.1291. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0d | method | Message ID |
| 4 | uint8 | type | Device name to set. Values:<br>• **0:** OTA device name |
| 5 | uint8array | name | Device name |

**Table 2.1292. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_set_device_name_rsp_t *gecko_cmd_system_set_device_name(uint8 type, uint8 name_len,
const uint8 *name_data);

/* Response id */
gecko_rsp_system_set_device_name_id

/* Response structure */
struct gecko_msg_system_set_device_name_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.12 cmd_system_set_identity_address

Set the device's Bluetooth identity address. The address can be a public device address or a static device address. A valid address set with this command will be written into persistent storage using PS keys. The stack returns an error if the static device address does not conform to the Bluetooth specification.

The new address will be effective in the next system reboot. The stack will use the address in the PS keys when present. Otherwise, it uses the default Bluetooth public device address which is programmed at production.

The stack treats 00:00:00:00:00:00 and ff:ff:ff:ff:ff:ff as invalid addresses. Therefore, passing one of them into this command will cause the stack to delete the PS keys and use the default address in the next system reboot.

**Note:** Because the PS keys are located in flash and flash wearing can occur, avoid calling this command regularly.

**Table 2.1293. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x13 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth identity address in little endian format |
| 10 | uint8 | type | Address type<br>• **0:** Public device address<br>• **1:** Static device address |

**Table 2.1294. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x13 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**BGLIB C API**

```
/* Function */
struct  gecko_msg_system_set_identity_address_rsp_t  *gecko_cmd_system_set_identity_address(bd_addr  address,
uint8 type);

/* Response id */
gecko_rsp_system_set_identity_address_id

/* Response structure */
struct gecko_msg_system_set_identity_address_rsp_t
{
  uint16 result;
};
```

#### 2.40.1.13 cmd_system_set_tx_power

Set the global maximum TX power for Bluetooth. The returned value is the selected maximum output power level after applying the RF path compensation. If the GATT server contains a TX power service, the TX Power Level attribute 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 is limited to 10 dBm if Adaptive Frequency Hopping (AFH) is not enabled.

By default, the global maximum TX power value is 8 dBm.

**NOTE:** Do not use this command while advertising, scanning, or during connection.

**Table 2.1295. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0a | method | Message ID |
| 4-5 | int16 | power | TX power in 0.1 dBm steps. For example, the value of 10 is 1 dBm and 55 is 5.5 dBm. |

**Table 2.1296. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x0a | method | Message ID |
| 4-5 | int16 | set_power | The selected maximum power level |

**BGLIB C API**

```
/* Function */
struct gecko_msg_system_set_tx_power_rsp_t *gecko_cmd_system_set_tx_power(int16 power);

/* Response id */
gecko_rsp_system_set_tx_power_id

/* Response structure */
struct gecko_msg_system_set_tx_power_rsp_t
{
  int16 set_power;
};
```

#### 2.40.2 system events

#### 2.40.2.1 evt_system_awake

Indicates that the device is awake and no longer in sleep mode.

**NOTE:** Stack does not generate this event by itself because sleep and wakeup are managed by applications. If this event is needed, call function gecko_send_system_awake(), which signals the stack to send this event.

**Table 2.1297. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x04 | method | Message ID |

**C Functions**

```
/* Event id */
gecko_evt_system_awake_id

/* Event structure */
struct gecko_msg_system_awake_evt_t
{
};
```

#### 2.40.2.2 evt_system_boot

Indicates that the device has started and the radio is ready. This event carries the firmware build number and other software and hardware identification codes.

**Table 2.1298. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x12 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | major | Major release version |
| 6-7 | uint16 | minor | Minor release version |
| 8-9 | uint16 | patch | Patch release number |
| 10-11 | uint16 | build | Build number |
| 12-15 | uint32 | bootloader | Bootloader version |
| 16-17 | uint16 | hw | Hardware type |
| 18-21 | uint32 | hash | Version hash |

**C Functions**

```
/* Event id */
gecko_evt_system_boot_id

/* Event structure */
struct gecko_msg_system_boot_evt_t
{
  uint16 major;,
  uint16 minor;,
  uint16 patch;,
  uint16 build;,
  uint32 bootloader;,
  uint16 hw;,
  uint32 hash;
};
```

#### 2.40.2.3 evt_system_error

Indicates that an error has occurred. See error codes table for more information.

**Table 2.1299. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | reason | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | data | Data related to the error; this field can be empty. |

**C Functions**

```
/* Event id */
gecko_evt_system_error_id

/* Event structure */
struct gecko_msg_system_error_evt_t
{
  uint16 reason;,
  uint8array data;
};
```

#### 2.40.2.4 evt_system_external_signal

Indicates that the external signals have been received. External signals are generated from the native application.

**Table 2.1300. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x03 | method | Message ID |
| 4-7 | uint32 | extsignals | Bitmask of external signals received since last event. |

**C Functions**

```
/* Event id */
gecko_evt_system_external_signal_id

/* Event structure */
struct gecko_msg_system_external_signal_evt_t
{
  uint32 extsignals;
};
```

#### 2.40.2.5 evt_system_hardware_error

Indicates that a hardware-related error has occurred.

**Table 2.1301. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | status | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |

**C Functions**

```
/* Event id */
gecko_evt_system_hardware_error_id

/* Event structure */
struct gecko_msg_system_hardware_error_evt_t
{
  uint16 status;
};
```

### 2.40.3 system enumerations

#### 2.40.3.1 enum_system_linklayer_config_key

These Keys are used to configure Link Layer Operation

**Table 2.1302. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | system_linklayer_config_key_halt | Same as system_halt command, value-0 Stop Radio 1-Start Radio |
| 2 | system_linklayer_config_key_priority_range | Sets the RAIL priority_mapping offset field of the link layer priority configuration structure to the first byte of the value field. |
| 3 | system_linklayer_config_key_scan_channels | Sets channels to scan on. The first byte of the value is the channel map. 0x1 = Channel 37, 0x2 = Channel 38, 0x4 = Channel 39 |
| 4 | system_linklayer_config_key_set_flags | Sets the link layer configuration flags. The value is a little endian 32-bit integer. Flag Values:<br>• 0x00000001 - Disable Feature Exchange when slave<br>• 0x00000002 - Disable Feature Exchange when master |
| 5 | system_linklayer_config_key_clr_flags | Value is flags to clear. Flags are the same as in SET_FLAGS command. |
| 7 | system_linklayer_config_key_set_afh_interval | Set afh_scan_interval field of Link Layer priority configuration structure. |
| 9 | system_linklayer_config_key_set_priority_table | Value contains priority table to be copied over the existing table. If value is smaller than full table then only those values are updated. see gecko_bluetooth_ll_priorities struct for the definition of priority table. |

## 2.41 Testing Commands (test)

### 2.41.1 test commands

#### 2.41.1.1 cmd_test_dtm_end

End a transmitter or a receiver test. When the command is processed by the radio and the test has ended, a test_dtm_completed event is triggered.

**Table 2.1303. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x02 | method | Message ID |

**Table 2.1304. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Command result |

**BGLIB C API**

```
/* Function */
struct gecko_msg_test_dtm_end_rsp_t *gecko_cmd_test_dtm_end();

/* Response id */
gecko_rsp_test_dtm_end_id

/* Response structure */
struct gecko_msg_test_dtm_end_rsp_t
{
  uint16 result;
};
```

**Table 2.1305. Events Generated**

| Event | Description |
|---|---|
| test_dtm_completed | Received when the command is processed by the radio and the test has ended. |

#### 2.41.1.2 cmd_test_dtm_rx

Start a receiver test against a separate Bluetooth tester device. When the command is processed by the radio, a test_dtm_completed event is triggered. This event indicates whether the test started successfully.

Parameter **phy** specifies which PHY is used to receive the packets. All devices support at least 1M PHY.

The test may be stopped using the test_dtm_end command. This will trigger another test_dtm_completed event, which carries the number of packets received during the test.

**Table 2.1306. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | channel | Bluetooth channel<br>**Range:** 0-39<br>Channel is (F - 2402) / 2,<br>where F is frequency in MHz |
| 5 | uint8 | phy | PHY to use |

**Table 2.1307. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Command result |

**BGLIB C API**

```
/* Function */
struct gecko_msg_test_dtm_rx_rsp_t *gecko_cmd_test_dtm_rx(uint8 channel, uint8 phy);

/* Response id */
gecko_rsp_test_dtm_rx_id

/* Response structure */
struct gecko_msg_test_dtm_rx_rsp_t
{
  uint16 result;
};
```

**Table 2.1308. Events Generated**

| Event | Description |
|---|---|
| test_dtm_completed | This event is received when the command is processed. |

#### 2.41.1.3 cmd_test_dtm_tx

Start a transmitter test against a separate Bluetooth tester device. When the command is processed by the radio, a test_dtm_completed event is triggered. This event indicates whether the test started successfully.

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. The parameter **phy** specifies which PHY is used to transmit the packets. All devices support at least 1M PHY. A special packet type, **test_pkt_carrier**, 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.

**Table 2.1309. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | packet_type | Packet type to transmit |
| 5 | uint8 | length | Packet length in bytes<br>**Range:** 0-255 |
| 6 | uint8 | channel | Bluetooth channel<br>**Range:** 0-39<br>Channel is (F - 2402) / 2,<br>where F is frequency in MHz |
| 7 | uint8 | phy | PHY to use |

**Table 2.1310. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Command result |

**BGLIB C API**

```
/* Function */
struct gecko_msg_test_dtm_tx_rsp_t *gecko_cmd_test_dtm_tx(uint8 packet_type, uint8 length, uint8 channel,
uint8 phy);

/* Response id */
gecko_rsp_test_dtm_tx_id

/* Response structure */
struct gecko_msg_test_dtm_tx_rsp_t
{
  uint16 result;
};
```

**Table 2.1311. Events Generated**

| Event | Description |
|---|---|
| test_dtm_completed | This event is received when the command is processed. |

### 2.41.2 test events

#### 2.41.2.1 evt_test_dtm_completed

Indicates that the radio has processed a test start or end command. The **result** parameter indicates the success of the command.

After the receiver or transmitter test is stopped, the **number_of_packets** parameter in this event indicates the number of received or transmitted packets.

**Table 2.1312. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing Commands |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Command result |
| 6-7 | uint16 | number_of_packets | Number of packets<br>Only valid for test_dtm_end command. |

**C Functions**

```
/* Event id */
gecko_evt_test_dtm_completed_id

/* Event structure */
struct gecko_msg_test_dtm_completed_evt_t
{
  uint16 result;,
  uint16 number_of_packets;
};
```

### 2.41.3 test enumerations

#### 2.41.3.1 enum_test_packet_type

Test packet types supported by the stack

**Table 2.1313. Enumerations**

| Value | Name | Description |
|---|---|---|
| 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 |

#### 2.41.3.2 enum_test_phy

Test PHY types

**Table 2.1314. Enumerations**

| Value | Name | Description |
|---|---|---|
| 1 | test_phy_1m | 1M PHY |
| 2 | test_phy_2m | 2M PHY |
| 3 | test_phy_125k | 125k Coded PHY |
| 4 | test_phy_500k | 500k Coded PHY |

## 2.42 User Messaging (user)

This class provides one command and one event which can be used by a NCP host and target to implement a communication mechanism with a custom proprietary protocol. An application must decide whether and how the command and event are used. The stack does not produce or consume any messages belonging to this class.

### 2.42.1 user commands

#### 2.42.1.1 cmd_user_message_to_target

Used by an NCP host to send a message to the target application on device. The application on target is must send the response with gecko_send_rsp_user_message_to_target.

**Table 2.1315. Command**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0xff | class | Message class: User Messaging |
| 3 | 0x00 | method | Message ID |
| 4 | uint8array | data | The message |

**Table 2.1316. Response**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0xff | class | Message class: User Messaging |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code<br>• **0:** success<br>• **Non-zero:** an error has occurred<br>For other values see Error codes |
| 6 | uint8array | data | The response message |

**BGLIB C API**

```
/* Function */
struct gecko_msg_user_message_to_target_rsp_t *gecko_cmd_user_message_to_target(uint8 data_len, const uint8
*data_data);

/* Response id */
gecko_rsp_user_message_to_target_id

/* Response structure */
struct gecko_msg_user_message_to_target_rsp_t
{
  uint16 result;,
  uint8array data;
};
```

### 2.42.2 user events

#### 2.42.2.1 evt_user_message_to_host

Used by the target application on a device to initiate communication and send a message to the NCP host. Do not send event messages in the context of the user command handling.

**Table 2.1317. Event**

| Byte | Type | Name | Description |
|---|---|---|---|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0xff | class | Message class: User Messaging |
| 3 | 0x00 | method | Message ID |
| 4 | uint8array | data | The message |

**C Functions**

```
/* Event id */
gecko_evt_user_message_to_host_id

/* Event structure */
struct gecko_msg_user_message_to_host_evt_t
{
  uint8array data;
};
```

### 2.43 Error codes

This chapter describes all BGAPI error codes.

- **Errors related to hardware**

| Code | Name | Description |
|---|---|---|
| 0x0501 | ps_store_full | Flash reserved for PS store is full |
| 0x0502 | ps_key_not_found | PS key not found |
| 0x0503 | i2c_ack_missing | Acknowledge for i2c was not received. |
| 0x0504 | i2c_timeout | I2C read or write timed out. |

- **Errors related to BGAPI protocol**

| Code | Name | Description |
|---|---|---|
| 0x0100 | success | No error |
| 0x0101 | invalid_conn_handle | Invalid GATT connection handle. |
| 0x0102 | waiting_response | Waiting response from GATT server to previous procedure. |
| 0x0103 | gatt_connection_timeout | GATT connection is closed due procedure timeout. |
| 0x0180 | invalid_param | Command contained invalid parameter |
| 0x0181 | wrong_state | Device is in wrong state to receive command |
| 0x0182 | out_of_memory | Device has run out of memory |
| 0x0183 | not_implemented | Feature is not implemented |

| Code | Name | Description |
|---|---|---|
| 0x0184 | invalid_command | Command was not recognized |
| 0x0185 | timeout | A command or procedure failed or a link lost due to timeout |
| 0x0186 | not_connected | Connection handle passed is to command is not a valid handle |
| 0x0187 | flow | Command would cause either underflow or overflow error |
| 0x0188 | user_attribute | User attribute was accessed through API which is not supported |
| 0x0189 | invalid_license_key | No valid license key found |
| 0x018a | command_too_long | Command maximum length exceeded |
| 0x018b | out_of_bonds | Bonding procedure can't be started because device has no space left for bond. |
| 0x018c | unspecified | Unspecified error |
| 0x018d | hardware | Hardware failure |
| 0x018e | buffers_full | Command not accepted, because internal buffers are full |
| 0x018f | disconnected | Command or Procedure failed due to disconnection |
| 0x0190 | too_many_requests | Too many Simultaneous Requests |
| 0x0191 | not_supported | Feature is not supported in this firmware build |
| 0x0192 | no_bonding | The bonding does not exist. |
| 0x0193 | crypto | Error using crypto functions |
| 0x0194 | data_corrupted | Data was corrupted. |
| 0x0195 | command_incomplete | Data received does not form a complete command |
| 0x0196 | not_initialized | Feature or subsystem not initialized |
| 0x0197 | invalid_sync_handle | Invalid periodic advertising sync handle |
| 0x0198 | invalid_module_action | Bluetooth cannot be used on this hardware |
| 0x0199 | radio | Error received from radio |

- **Errors from Security Manager Protocol**

| Code | Name | Description |
|---|---|---|
| 0x0301 | passkey_entry_failed | The user input of passkey failed, for example, the user cancelled the operation |
| 0x0302 | oob_not_available | Out of Band data is not available for authentication |
| 0x0303 | authentication_requirements | The pairing procedure cannot be performed as authentication requirements cannot be met due to IO capabilities of one or both devices |
| 0x0304 | confirm_value_failed | The confirm value does not match the calculated compare value |
| 0x0305 | pairing_not_supported | Pairing is not supported by the device |
| 0x0306 | encryption_key_size | The resultant encryption key size is insufficient for the security requirements of this device |

| Code | Name | Description |
|---|---|---|
| 0x0307 | command_not_supported | The SMP command received is not supported on this device |
| 0x0308 | unspecified_reason | Pairing failed due to an unspecified reason |
| 0x0309 | repeated_attempts | Pairing or authentication procedure is disallowed because too little time has elapsed since last pairing request or security request |
| 0x030a | invalid_parameters | The Invalid Parameters error code indicates: the command length is invalid or a parameter is outside of the specified range. |
| 0x030b | dhkey_check_failed | Indicates to the remote device that the DHKey Check value received doesn't match the one calculated by the local device. |
| 0x030c | numeric_comparison_failed | Indicates that the confirm values in the numeric comparison protocol do not match. |
| 0x030d | bredr_pairing_in_progress | Indicates that the pairing over the LE transport failed due to a Pairing Request sent over the BR/EDR transport in process. |
| 0x030e | cross_transport_key_derivation_generation_not_allowed | 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. |

- **Bluetooth errors**

| Code | Name | Description |
|---|---|---|
| 0x0200 | error_success | Command completed succesfully |
| 0x0202 | unknown_connection_identifier | Connection does not exist, or connection open request was cancelled. |
| 0x0205 | authentication_failure | 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 | pin_or_key_missing | Pairing failed because of missing PIN, or authentication failed because of missing Key |
| 0x0207 | memory_capacity_exceeded | Controller is out of memory. |
| 0x0208 | connection_timeout | Link supervision timeout has expired. |
| 0x0209 | connection_limit_exceeded | Controller is at limit of connections it can support. |
| 0x020a | synchronous_connectiontion_limit_exceeded | 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 | acl_connection_already_exists | 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_disallowed | Command requested cannot be executed because the Controller is in a state where it cannot process this command at this time. |
| 0x020d | connection_rejected_due_to_limited_resources | The Connection Rejected Due To Limited Resources error code indicates that an incoming connection was rejected due to limited resources. |

| Code | Name | Description |
|---|---|---|
| 0x020e | connection_rejected_due_to_security_reasons | 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 | connection_rejected_due_to_unacceptable_bd_addr | 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 | connection_accept_timeout_exceeded | The Connection Accept Timeout has been exceeded for this connection attempt. |
| 0x0211 | unsupported_feature_or_parameter_value | A feature or parameter value in the HCI command is not supported. |
| 0x0212 | invalid_command_parameters | Command contained invalid parameters. |
| 0x0213 | remote_user_terminated | User on the remote device terminated the connection. |
| 0x0214 | remote_device_terminated_connection_due_to_low_resources | The remote device terminated the connection because of low resources |
| 0x0215 | remote_powering_off | Remote Device Terminated Connection due to Power Off |
| 0x0216 | connection_terminated_by_local_host | Local device terminated the connection. |
| 0x0217 | repeated_attempts | The Controller is disallowing an authentication or pairing procedure because too little time has elapsed since the last authentication or pairing attempt failed. |
| 0x0218 | pairing_not_allowed | The device does not allow pairing. This can be for example, when a device only allows pairing during a certain time window after some user input allows pairing |
| 0x021a | unsupported_remote_feature | The remote device does not support the feature associated with the issued command. |
| 0x021f | unspecified_error | No other error code specified is appropriate to use. |
| 0x0222 | ll_response_timeout | Connection terminated due to link-layer procedure timeout. |
| 0x0223 | ll_procedure_collision | LL procedure has collided with the same transaction or procedure that is already in progress. |
| 0x0225 | encryption_mode_not_acceptable | The requested encryption mode is not acceptable at this time. |
| 0x0226 | link_key_cannot_be_changed | Link key cannot be changed because a fixed unit key is being used. |
| 0x0228 | instant_passed | LMP PDU or LL PDU that includes an instant cannot be performed because the instant when this would have occurred has passed. |
| 0x0229 | pairing_with_unit_key_not_supported | It was not possible to pair as a unit key was requested and it is not supported. |
| 0x022a | different_transaction_collision | LMP transaction was started that collides with an ongoing transaction. |
| 0x022e | channel_assessment_not_supported | The Controller cannot perform channel assessment because it is not supported. |
| 0x022f | insufficient_security | The HCI command or LMP PDU sent is only possible on an encrypted link. |

| Code | Name | Description |
|---|---|---|
| 0x0230 | parameter_out_of_mandatory_range | A parameter value requested is outside the mandatory range of parameters for the given HCI command or LMP PDU. |
| 0x0237 | simple_pairing_not_supported_by_host | 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 | host_busy_pairing | The Host is busy with another pairing operation and unable to support the requested pairing. The receiving device should retry pairing again later. |
| 0x0239 | connection_rejected_due_to_no_suitable_channel_found | The Controller could not calculate an appropriate value for the Channel selection operation. |
| 0x023a | controller_busy | Operation was rejected because the controller is busy and unable to process the request. |
| 0x023b | unacceptable_connection_interval | Remote device terminated the connection because of an unacceptable connection interval. |
| 0x023c | advertising_timeout | Ddvertising for a fixed duration completed or, for directed advertising, that advertising completed without a connection being created. |
| 0x023d | connection_terminated_due_to_mic_failure | Connection was terminated because the Message Integrity Check (MIC) failed on a received packet. |
| 0x023e | connection_failed_to_be_established | LL initiated a connection but the connection has failed to be established. Controller did not receive any packets from remote end. |
| 0x023f | mac_connection_failed | The MAC of the 802.11 AMP was requested to connect to a peer, but the connection failed. |
| 0x0240 | coarse_clock_adjustment_rejected_but_will_try_to_adjust_using_clock_dragging | 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 | unknown_advertising_identifier | 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 | limit_reached | Number of operations requested has been reached and has indicated the completion of the activity (e.g., advertising or scanning). |
| 0x0244 | operation_cancelled_by_host | A request to the Controller issued by the Host and still pending was successfully canceled. |
| 0x0245 | packet_too_long | An attempt was made to send or receive a packet that exceeds the maximum allowed packet length. |

- **Application errors**

| Code | Name | Description |
|---|---|---|
| 0x0a01 | file_open_failed | File open failed. |
| 0x0a02 | xml_parse_failed | XML parsing failed. |
| 0x0a03 | device_connection_failed | Device connection failed. |
| 0x0a04 | device_comunication_failed | Device communication failed. |

| Code | Name | Description |
|---|---|---|
| 0x0a05 | authentication_failed | Device authentication failed. |
| 0x0a06 | incorrect_gatt_database | Device has incorrect GATT database. |
| 0x0a07 | disconnected_due_to_procedure_collision | Device disconnected due to procedure collision. |
| 0x0a08 | disconnected_due_to_secure_session_failed | Device disconnected due to failure to establish or rees-tablish a secure session. |
| 0x0a09 | encryption_decryption_error | Encrypion/decryption operation failed. |
| 0x0a0a | maximum_retries | Maximum allowed retries exceeded. |
| 0x0a0b | data_parse_failed | Data parsing failed. |
| 0x0a0c | pairing_removed | Pairing established by the application layer protocol has been removed. |
| 0x0a0d | inactive_timeout | Inactive timeout. |
| 0x0a0e | mismatched_or_insufficient_security | Mismatched or insufficient security level |

- **Errors from Attribute Protocol**

| Code | Name | Description |
|---|---|---|
| 0x0401 | invalid_handle | The attribute handle given was not valid on this server |
| 0x0402 | read_not_permitted | The attribute cannot be read |
| 0x0403 | write_not_permitted | The attribute cannot be written |
| 0x0404 | invalid_pdu | The attribute PDU was invalid |
| 0x0405 | insufficient_authentication | The attribute requires authentication before it can be read or written. |
| 0x0406 | request_not_supported | Attribute Server does not support the request received from the client. |
| 0x0407 | invalid_offset | Offset specified was past the end of the attribute |
| 0x0408 | insufficient_authorization | The attribute requires authorization before it can be read or written. |
| 0x0409 | prepare_queue_full | Too many prepare writes have been queueud |
| 0x040a | att_not_found | No attribute found within the given attribute handle range. |
| 0x040b | att_not_long | The attribute cannot be read or written using the Read Blob Request |
| 0x040c | insufficient_enc_key_size | The Encryption Key Size used for encrypting this link is insufficient. |
| 0x040d | invalid_att_length | The attribute value length is invalid for the operation |
| 0x040e | unlikely_error | The attribute request that was requested has encoun-tered an error that was unlikely, and therefore could not be completed as requested. |
| 0x040f | insufficient_encryption | The attribute requires encryption before it can be read or written. |
| 0x0410 | unsupported_group_type | The attribute type is not a supported grouping attribute as defined by a higher layer specification. |
| 0x0411 | insufficient_resources | Insufficient Resources to complete the request |

| Code | Name | Description |
|---|---|---|
| 0x0412 | out_of_sync | The server requests the client to rediscover the database. |
| 0x0413 | value_not_allowed | The attribute parameter value was not allowed. |
| 0x0480 | application | Start of ATT application error codes range defined by a higher layer specification. |

- **Bluetooth Mesh errors**

| Code | Name | Description |
|---|---|---|
| 0x0c01 | already_exists | Returned when trying to add a key or some other unique resource with an ID which already exists |
| 0x0c02 | does_not_exist | Returned when trying to manipulate a key or some other resource with an ID which does not exist |
| 0x0c03 | limit_reached | Returned when an operation cannot be executed because a pre-configured limit for keys, key bindings, elements, models, virtual addresses, provisioned devices, or provisioning sessions is reached |
| 0x0c04 | invalid_address | Returned when trying to use a reserved address or add a "pre-provisioned" device using an address already used by some other device |
| 0x0c05 | malformed_data | In a BGAPI response, the user supplied malformed data; in a BGAPI event, the remote end responded with malformed or unrecognized data |
| 0x0c06 | already_initialized | An attempt was made to initialize a subsystem that was already initialized. |
| 0x0c07 | not_initialized | An attempt was made to use a subsystem that wasn't initialized yet. Call the subsystem's init function first. |
| 0x0c08 | no_friend_offer | Returned when trying to establish a friendship as a Low Power Node, but no acceptable friend offer message was received. |
| 0x0c09 | prov_link_closed | Provisioning link was unexpectedly closed before provisioning was complete. |
| 0x0c0a | prov_invalid_pdu | An unrecognized provisioning PDU was received. |
| 0x0c0b | prov_invalid_pdu_format | A provisioning PDU with wrong length or containing field values that are out of bounds was received. |
| 0x0c0c | prov_unexpected_pdu | An unexpected (out of sequence) provisioning PDU was received. |
| 0x0c0d | prov_confirmation_failed | The computed confirmation value did not match the expected value. |
| 0x0c0e | prov_out_of_resources | Provisioning could not be continued due to insufficient resources. |
| 0x0c0f | prov_decryption_failed | The provisioning data block could not be decrypted. |
| 0x0c10 | prov_unexpected_error | An unexpected error happened during provisioning. |
| 0x0c11 | prov_cannot_assign_addr | Device could not assign unicast addresses to all of its elements. |
| 0x0c12 | address_temporarily_unavailable | Returned when trying to reuse an address of a previously deleted device before an IV Index Update has been executed. |

| Code | Name | Description |
|---|---|---|
| 0x0c13 | address_already_used | Returned when trying to assign an address that is used by one of the devices in the Device Database, or by the Provisioner itself. |
| 0x0c14 | no_data_available | Returned when no data is available for reading. |

- **Bluetooth Mesh foundation errors**

| Code | Name | Description |
|---|---|---|
| 0x0e01 | invalid_address | Returned when address in request was not valid |
| 0x0e02 | invalid_model | Returned when model identified is not found for a given element |
| 0x0e03 | invalid_app_key | Returned when the key identified by AppKeyIndex is not stored in the node |
| 0x0e04 | invalid_net_key | Returned when the key identified by NetKeyIndex is not stored in the node |
| 0x0e05 | insufficient_resources | Returned when The node cannot serve the request due to insufficient resources |
| 0x0e06 | key_index_exists | Returned when the key identified is already stored in the node and the new NetKey value is different |
| 0x0e07 | invalid_publish_params | Returned when the model does not support the publish mechanism |
| 0x0e08 | not_subscribe_model | Returned when the model does not support the subscribe mechanism |
| 0x0e09 | storage_failure | Returned when storing of the requested parameters failed |
| 0x0e0a | not_supported | Returned when requested setting is not supported |
| 0x0e0b | cannot_update | Returned when the requested update operation cannot be performed due to general constraints |
| 0x0e0c | cannot_remove | Returned when the requested delete operation cannot be performed due to general constraints |
| 0x0e0d | cannot_bind | Returned when the requested bind operation cannot be performed due to general constraints |
| 0x0e0e | temporarily_unable | Returned when The node cannot start advertising with Node Identity or Proxy since the maximum number of parallel advertising is reached |
| 0x0e0f | cannot_set | Returned when the requested state cannot be set |
| 0x0e10 | unspecified | Returned when an unspecified error took place |
| 0x0e11 | invalid_binding | Returned when the NetKeyIndex and AppKeyIndex combination is not valid for a Config AppKey Update |

- **Filesystem errors**

| Code | Name | Description |
|---|---|---|
| 0x0901 | file_not_found | File not found |

- **Errors from Logical Link Control and Adaptation Protocol**

| Code | Name | Description |
|---|---|---|
| 0x0d01 | remote_disconnected | Returned when remote disconnects the connection-oriented channel by sending disconnection request. |
| 0x0d02 | local_disconnected | Returned when local host disconnect the connection-oriented channel by sending disconnection request. |
| 0x0d03 | cid_not_exist | Returned when local host did not find a connection-oriented channel with given destination CID. |
| 0x0d04 | le_disconnected | Returned when connection-oriented channel disconnected due to LE connection is dropped. |
| 0x0d05 | flow_control_violated | Returned when connection-oriented channel disconnected due to remote end send data even without credit. |
| 0x0d06 | flow_control_credit_overflowed | Returned when connection-oriented channel disconnected due to remote end send flow control credits exceed 65535. |
| 0x0d07 | no_flow_control_credit | Returned when connection-oriented channel has run out of flow control credit and local application still trying to send data. |
| 0x0d08 | connection_request_timeout | Returned when connection-oriented channel has not received connection response message within maximum timeout. |
| 0x0d09 | invalid_cid | Returned when local host received a connection-oriented channel connection response with an invalid destination CID. |
| 0x0d0a | wrong_state | Returned when local host application tries to send a command which is not suitable for L2CAP channel's current state. |

- **Security errors**

| Code | Name | Description |
|---|---|---|
| 0x0b01 | image_signature_verification_failed | Device firmware signature verification failed. |
| 0x0b02 | file_signature_verification_failed | File signature verification failed. |
| 0x0b03 | image_checksum_error | Device firmware checksum is not valid. |

# 3. Document Revision History

**Table 3.1. Document Revision History**

| Revision Number | Effective Date | Change Description |
|---|---|---|
| 1.0 | April 1st 2015 | Initial version. |
| 1.1 | December 23rd 2015 | Updated for firmware version 0.9.2. |
| 1.2 | January 15th 2016 | Corrected typography and formatting issues. |
| 1.3 | February 12th 2016 | Updated for firmware version 1.0.0. |
| 1.4 | March 24th 2016 | Updated for firmware version 1.0.2. |
| 1.5 | June 6th 2016 | Updated for firmware version 1.0.4. |
| 1.6 | June 15th 2016 | Revised description for timestamp parameter in evt_hardware_interrupt. |
| 1.7 | September 2nd 2016 | Updated for firmware version 2.0.0. |
| 1.8 | October 13th 2016 | Corrected default ATT MTU value. |
| 1.9 | December 2nd 2016 | Updated for firmware version 2.1.0. |
| 2.0 | March 10th 2017 | Updated for firmware version 2.3.0. |
| 2.1 | July 9th 2017 | Updated for firmware version 2.4.0. |
| 2.1.1 | July 17th, 2017 | Updated for Bluetooth Mesh Beta. |
| 2.2 | February 23rd, 2018 | Updated for Bluetooth Mesh 1.2.0 GA with Bluetooth firmware version 2.7.1 |
| 2.3 | July 13th, 2018 | Updated for Bluetooth Mesh 1.3.0 GA with Bluetooth firmware version 2.9.1 |
| 2.4 | August 29th, 2018 | Updated for Bluetooth Mesh 1.3.1 GA with Bluetooth firmware version 2.9.3 |
| 2.5 | September 4th, 2018 | Updated for Bluetooth Mesh 1.3.2 GA with Bluetooth firmware version 2.9.3 |
| 2.6 | September 18th, 2018 | Updated for Bluetooth Mesh 1.3.3 GA with Bluetooth firmware version 2.9.3 |
| 2.7 | October 5th, 2018 | Updated for Bluetooth Mesh 1.3.4 GA with Bluetooth firmware version 2.9.3 |
| 2.8 | December 21st, 2018 | Updated for Bluetooth Mesh 1.4.0 GA with Bluetooth firmware version 2.11.0 |
| 2.9 | February 27th, 2019 | Updated for Bluetooth Mesh 1.4.1 GA with Bluetooth firmware version 2.11.2 |
| 2.10 | March 25th, 2019 | Updated for Bluetooth Mesh 1.4.2 GA with Bluetooth firmware version 2.11.3 |
| 2.11 | April 29th, 2019 | Updated for Bluetooth Mesh 1.4.3 GA with Bluetooth firmware version 2.11.4 |
| 2.12 | June 7th, 2019 | Updated for Bluetooth Mesh 1.5.0 GA with Bluetooth firmware version 2.12.0 |
| 2.13 | July 31st, 2019 | Updated for Bluetooth Mesh 1.5.1 GA with Bluetooth firmware version 2.12.1 |
| 2.14 | September 16th, 2019 | Updated for Bluetooth Mesh 1.5.2 GA with Bluetooth firmware version 2.12.3 |
| 2.15 | October 10th, 2019 | Updated for Bluetooth Mesh 1.6.0 Alpha with Bluetooth firmware version 2.13.0 Alpha |
| 2.16 | November 19th, 2019 | Updated for Bluetooth Mesh 1.6.0 Beta with Bluetooth firmware version 2.13.0 Beta |
| 2.17 | December 2nd, 2019 | Updated for Bluetooth Mesh 1.5.3 with Bluetooth firmware version 2.12.4 |
| 2.18 | December 20th, 2019 | Updated for Bluetooth Mesh 1.6.0 with Bluetooth firmware version 2.13.0 |

| Revision Number | Effective Date | Change Description |
|---|---|---|
| 2.19 | February 3rd, 2020 | Updated for Bluetooth Mesh 1.6.1 with Bluetooth firmware version 2.13.1 |
| 2.20 | March 15th, 2020 | Updated for Bluetooth Mesh 1.6.2 with Bluetooth firmware version 2.13.2 |
| 2.21 | April 6th, 2020 | Updated for Bluetooth Mesh 1.5.4 with Bluetooth firmware version 2.12.4 |
| 2.22 | May 8th, 2020 | Updated for Bluetooth Mesh 1.6.3 with Bluetooth firmware version 2.13.4 |
| 2.23 | June 12th, 2020 | Updated for Bluetooth Mesh 1.7.0 with Bluetooth firmware version 2.13.6 |
| 2.24 | August 27th, 2020 | Updated for Bluetooth Mesh 1.7.1 with Bluetooth firmware version 2.13.7 |
| 2.25 | October 28th, 2020 | Updated for Bluetooth Mesh 1.7.2 with Bluetooth firmware version 2.13.8 |
| 2.26 | December 13th, 2020 | Updated for Bluetooth Mesh 1.7.3 with Bluetooth firmware version 2.13.8 |
| 2.27 | March 1st, 2021 | Updated for Bluetooth Mesh 1.7.4 with Bluetooth firmware version 2.13.9 |
| 2.28 | August 13th, 2021 | Updated for Bluetooth Mesh 1.7.5 with Bluetooth firmware version 2.13.10 |

# Smart. Connected. Energy-Friendly.


**IoT Portfolio**
www.silabs.com/products

**Quality**
www.silabs.com/quality

**Support & Community**
www.silabs.com/community



SILICON LABS

**Silicon Laboratories Inc.**
**400 West Cesar Chavez**
**Austin, TX 78701**
**USA**

**www.silabs.com**