Bluetooth Mesh Software API Reference Manual

This document contains the full API reference for the Silicon Labs Bluetooth Mesh Software version 1.3.0, based on the Bluetooth LE Software version 2.9.1.

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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 the 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 This value is normally received from the gatt_service event.
characteristic	2 bytes	GATT characteristic handle 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 bytes Universal Unique Identifier (UUID)
att_errorcode	1 byte	Attribute protocol error code • **0:** No error • **Non-zero:** See Bluetooth specification, Host volume, Attribute Protocol, Error Codes table.
att_opcode	1 byte	Attribute opcode which informs the procedure from which attribute the value was received
uuid_128	16 bytes	128-bit UUID
aes_key_128	16 bytes	128-bit AES Key

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 and the actual data in 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 will return error code 0x018a (command_too_long) if the payload will cause an overflow.

Deprecation Notice

Note that some commands, enumerations and events are marked as deprecated. The usage of those commands is not recommended anymore as they will be removed in the future releases.

2.1 Cooexistence interface (coex)

Coexistence BGAPI class. Coexistence interface is enabled and initialised with gecko_initCoexHAL() function. Interface is configured with HAL configurator.

2.1.1 coex commands

2.1.1.1 cmd_coex_get_counters

This command is used to 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, high priority tx aborted.

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: Cooexistence interface
3	0x01	method	Message ID
4	uint8	reset	Reset counter values

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: Cooexistence interface
3	0x01	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the
6	uint8array	counters	Variable length byte array. The first byte defines the length of the data that follows, 0 - 255 bytes.

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_options

This command is used to configure coexistence options at runtime.

Table 2.3. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x08	lolen	Minimum payload length
2	0x20	class	Message class: Cooexistence 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 mask parameter.

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: Cooexistence interface
3	0x00	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.1.2 coex enumerations

2.1.2.1 enum_coex_option

Coexistence configuration options

Table 2.5. 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 Device Firmware Upgrade (dfu)

These commands and events are related to controlling firmware update over the configured host interface and are available only when the device has been booted into DFU mode. **The 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 the data has been uploaded
5. Send when all the data has been uploaded
6. Finalize the 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 with CRC checksum.

2.2.1 dfu commands

2.2.1.1 cmd_dfu_flash_set_address

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

Table 2.6. 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 2.7. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.2.1.2 cmd_dfu_flash_upload

This command can be used to upload the whole firmware image file into the Bluetooth device. The passed data length must be a multiple of 4 bytes. As 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. The next address of the flash sector in memory to write to is automatically updated by the bootloader after each individual command.

Table 2.8. 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 2.9. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.2.1.3 cmd_dfu_flash_upload_finish

This command can be used to tell to the device that the DFU file has been fully uploaded. To return the device back to normal mode the command DFU Reset must be issued next.

Table 2.10. 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 2.11. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.2.1.4 cmd_dfu_reset

This command can be used to reset the system. This 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 2.12. 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: • **0:** Normal reset • **1:** Boot to UART DFU mode • **2:** Boot to OTA DFU mode

BGLIB C API

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

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

Table 2.13. Events Generated

Event	Description
system_boot	Sent after the device has booted into normal mode
dfu_boot	Sent after the device has booted into UART DFU mode

2.2.2 dfu events

2.2.2.1 evt_dfu_boot

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

Table 2.14. 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;
};
```

2.2.2.2 evt_dfu_boot_failure

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

Table 2.15. 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, refer to the Error codes

C Functions

```
/* Event id */
gecko_evt_dfu_boot_failure_id

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

2.3 Endpoint (endpoint)

This class provides a command for closing Bluetooth connections.

2.3.1 endpoint commands

2.3.1.1 (deprecated) cmd_endpoint_close

Deprecated. Use new command le_connection_close to close Bluetooth connections.

This command can be used to close a Bluetooth connection. The parameter is a connection handle which is reported in event le_connection_opened.

Table 2.16. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x01	lolen	Minimum payload length
2	0x0b	class	Message class: Endpoint
3	0x02	method	Message ID
4	uint8	endpoint	The connection handle

Table 2.17. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x03	lolen	Minimum payload length
2	0x0b	class	Message class: Endpoint
3	0x02	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8	endpoint	The connection handle that was closed

BGLIB C API

```
/* Function */
struct gecko_msg_endpoint_close_rsp_t *gecko_cmd_endpoint_close(uint8 endpoint);

/* Response id */
gecko_rsp_endpoint_close_id

/* Response structure */
struct gecko_msg_endpoint_close_rsp_t
{
  uint16 result;,
  uint8 endpoint;
};
```

Table 2.18. Events Generated

Event	Description
le_connection_closed	This event indicates that a connection was closed.

2.4 Persistent Store (flash)

Persistent Store commands can be used to 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. As 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.

2.4.1 flash commands

2.4.1.1 cmd_flash_ps_erase

This command can be used to erase a single PS key and its value from the persistent store..

Table 2.19. 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 erase

Table 2.20. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.4.1.2 cmd_flash_ps_erase_all

This command can be used to erase all PS keys and their corresponding values.

Table 2.21. 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 2.22. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.4.1.3 cmd_flash_ps_load

This command can be used for retrieving the value of the specified PS key.

Table 2.23. 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 2.24. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.4.1.4 cmd_flash_ps_save

This command can be used to 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) will be returned if more than 56 bytes data is passed in.

Table 2.25. 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 2.26. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_d
ata);

/* Response id */
gecko_rsp_flash_ps_save_id

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

2.5 Generic Attribute Profile (gatt)

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

2.5.1 gatt commands

2.5.1.1 cmd_gatt_discover_characteristics

This command can be used to discover all characteristics of the defined 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 has succesfully completed or failed with error.

Table 2.27. 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 This value is normally received from the gatt_service event.

Table 2.28. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.29. Events Generated

Event	Description
gatt_characteristic	Discovered characteristic from remote GATT database.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.2 cmd_gatt_discover_characteristics_by_uuid

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

Table 2.30. 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 This value is normally received from the gatt_service event.
9	uint8array	uuid	Characteristic UUID

Table 2.31. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_discover_characteristics_by_uuid_rsp_t *gecko_cmd_gatt_discover_characteristics_by_uuid(u
int8 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 2.32. Events Generated

Event	Description
gatt_characteristic	Discovered characteristic from remote GATT database.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.3 cmd_gatt_discover_descriptors

This command can be used to discover all the descriptors of the specified remote GATT characteristics in a remote GATT database. This command generates a unique gatt_descriptor event for every discovered descriptor. Received gatt_procedure_completed event indicates that this GATT procedure has succesfully completed or failed with error.

Table 2.33. 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 This value is normally received from the gatt_characteristic event.

Table 2.34. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.35. Events Generated

Event	Description
gatt_descriptor	Discovered descriptor from remote GATT database.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.4 cmd_gatt_discover_primary_services

This command can be used to discover all the 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 error.

Table 2.36. 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 2.37. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_discover_primary_services_rsp_t *gecko_cmd_gatt_discover_primary_services(uint8 connectio
n);

/* Response id */
gecko_rsp_gatt_discover_primary_services_id

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

Table 2.38. Events Generated

Event	Description
gatt_service	Discovered service from remote GATT database
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.5 cmd_gatt_discover_primary_services_by_uuid

This command can be used to 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 has succesfully completed or failed with error.

Table 2.39. 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

Table 2.40. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct  gecko_msg_gatt_discover_primary_services_by_uuid_rsp_t  *gecko_cmd_gatt_discover_primary_services_by_uui
d(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 2.41. Events Generated

Event	Description
gatt_service	Discovered service from remote GATT database.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.6 cmd_gatt_execute_characteristic_value_write

This command can be used to commit or cancel previously queued writes to a long characteristic of a remote GATT server. Writes are sent to queue with 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 has been written successfully or that an error response has been received.

Table 2.42. 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	Unsigned 8-bit integer

Table 2.43. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_execute_characteristic_value_write_rsp_t *gecko_cmd_gatt_execute_characteristic_value_wri
te(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 2.44. Events Generated

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.7 cmd_gatt_find_included_services

This command can be used to find out if a service of a remote GATT database includes one or more other services. This command generates a unique gatt_service_completed event for each included service. This command generates a unique gatt_service event for every discovered service. Received gatt_procedure_completed event indicates that this GATT procedure has successfully completed or failed with error.

Table 2.45. 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 This value is normally received from the gatt_service event.

Table 2.46. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_find_included_services_rsp_t *gecko_cmd_gatt_find_included_services(uint8 connection, uin
t32 service);

/* Response id */
gecko_rsp_gatt_find_included_services_id

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

Table 2.47. Events Generated

Event	Description
gatt_service	Discovered service from remote GATT database.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.8 cmd_gatt_prepare_characteristic_value_reliable_write

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

Table 2.48. 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 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 2.49. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 *valu
e_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 2.50. Events Generated

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.9 cmd_gatt_prepare_characteristic_value_write

This command can be used to add a characteristic value to the write queue of a remote GATT server. This command can be used in cases where very long attributes need to be written, or a set of values needs to be written atomically. At most ATT_MTU - 5 amount of data can be sent once. Writes are executed or cancelled with the execute_characteristic_value_write command. Whether the writes succeeded or not are indicated in the response of the execute_characteristic_value_write command.

In all cases where the amount of data to transfer fits into the BGAPI payload the command gatt_write_characteristic_value is recommended for writing long values since it transparently performs the prepare_write and execute_write commands.

Table 2.51. 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 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 2.52. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_wri
te(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 2.53. Events Generated

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.10 cmd_gatt_read_characteristic_value

This command can be used to 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 events are generated because the firmware will automatically use the "read long" GATT procedure. A received gatt_procedure_completed event indicates that all data has been read successfully or that an error response has been received.

Note that the GATT client does not verify if the requested atrribute is a characteristic value. Thus before calling this command the application should make sure the attribute handle is for a characteristic value in some means, for example, by performing characteristic discovery.

Table 2.54. 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 This value is normally received from the gatt_characteristic event.

Table 2.55. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_read_characteristic_value_rsp_t *gecko_cmd_gatt_read_characteristic_value(uint8 connectio
n, 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 2.56. Events Generated

Event	Description
gatt_characteristic_value	This event contains the data belonging to a characteristic sent by the GATT Server.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.11 cmd_gatt_read_characteristic_value_by_uuid

This command can be used to read the characteristic value of a service from a remote GATT database by giving the UUID of the characteristic and the handle of the service containing this characteristic. A single gatt_characteristic_value event is generated if the characteristic value fits in one ATT PDU. Otherwise more than one gatt_characteristic_value events are generated because the firmware will automatically use the "read long" GATT procedure. A received gatt_procedure_completed event indicates that all data has been read successfully or that an error response has been received.

Table 2.57. 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 This value is normally received from the gatt_service event.
9	uint8array	uuid	Characteristic UUID

Table 2.58. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct  gecko_msg_gatt_read_characteristic_value_by_uuid_rsp_t  *gecko_cmd_gatt_read_characteristic_value_by_uui
d(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 2.59. Events Generated

Event	Description
gatt_characteristic_value	This event contains the data belonging to a characteristic sent by the GATT Server.

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.12 cmd_gatt_read_characteristic_value_from_offset

This command can be used to read a partial characteristic value with specified offset and maximum length from a remote GATT database. It is equivalent to 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" GATT procedure. A received gatt_procedure_completed event indicates that all data has been read successfully or that an error response has been received.

Table 2.60. 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 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 value starting at given offset will be read.

Table 2.61. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_read_characteristic_value_from_offset_rsp_t *gecko_cmd_gatt_read_characteristic_value_fro
m_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 2.62. Events Generated

Event	Description
gatt_characteristic_value	This event contains the data belonging to a characteristic sent by the GATT Server.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.13 cmd_gatt_read_descriptor_value

This command can be used to 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 will automatically use the "read long" GATT procedure. A received gatt_procedure_completed event indicates that all data has been read successfully or that an error response has been received.

Table 2.63. 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 2.64. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_read_descriptor_value_rsp_t *gecko_cmd_gatt_read_descriptor_value(uint8 connection, uint1
6 descriptor);

/* Response id */
gecko_rsp_gatt_read_descriptor_value_id

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

Table 2.65. Events Generated

Event	Description
gatt_descriptor_value	Descriptor value received from the remote GATT server.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.14 cmd_gatt_read_multiple_characteristic_values

This command can be used to read the values of multiple characteristics from a remote GATT database at once. gatt_characteristic_value events are generated as the values are returned by the remote GATT server. A received gatt_procedure_completed event indicates that either all data has been read successfully or that an error response has been received.

Table 2.66. 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	Little endian encoded uint16 list of characteristics to be read.

Table 2.67. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_read_multiple_characteristic_values_rsp_t *gecko_cmd_gatt_read_multiple_characteristic_va
lues(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 2.68. Events Generated

Event	Description
gatt_characteristic_value	This event contains the data belonging to a characteristic sent by the GATT Server.
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.15 cmd_gatt_send_characteristic_confirmation

This command must be used to send a characteristic confirmation to a remote GATT server after receiving an indication. The gatt_characteristic_value_event carries the att_opcode containing handle_value_indication (0x1d) which reveals that an indication has been received and this must be confirmed with this command. Confirmation needs to be sent within 30 seconds, otherwise the GATT transactions between the client and the server are discontinued.

Table 2.69. 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 2.70. 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	0x0d	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_send_characteristic_confirmation_rsp_t *gecko_cmd_gatt_send_characteristic_confirmation(u
int8 connection);

/* Response id */
gecko_rsp_gatt_send_characteristic_confirmation_id

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

2.5.1.16 cmd_gatt_set_characteristic_notification

This command can be used to enable or disable the notifications and indications being sent from a remote GATT server. This procedure discovers a characteristic client configuration descriptor and writes the related configuration flags to a remote GATT database. A received gatt_procedure_completed event indicates that this GATT procedure has successfully completed or that is has failed with an error.

Table 2.71. 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 This value is normally received from the gatt_characteristic event.
7	uint8	flags	Characteristic client configuration flags

Table 2.72. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_set_characteristic_notification_rsp_t *gecko_cmd_gatt_set_characteristic_notification(uin
t8 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 2.73. Events Generated

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with 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 on the GATT server side are defined by an upper level, for example by a profile; **so it is possible that no values are ever received, or that it may take time, depending on how the server is configured.**

2.5.1.17 cmd_gatt_set_max_mtu

This command can be used to set the maximum size of ATT Message Transfer Units (MTU). If the given value is too large according to the maximum BGAPI payload size, the system will select the maximal possible value as the maximum ATT_MTU. If maximum ATT_MTU is larger than 23, MTU is exchanged automatically after a Bluetooth connection has been established.

Table 2.74. 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 • Range: 23 to 250 Default: 247

Table 2.75. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_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;
};
```

2.5.1.18 cmd_gatt_write_characteristic_value

This command can be used to write the value of a characteristic in a remote GATT database. If the given value does not fit in one ATT PDU, "write long" GATT procedure is used automatically. Received gatt_procedure_completed event indicates that all data has been written successfully or that an error response has been received.

Table 2.76. 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 This value is normally received from the gatt_characteristic event.
7	uint8array	value	Characteristic value

Table 2.77. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_write_characteristic_value_rsp_t *gecko_cmd_gatt_write_characteristic_value(uint8 connect
ion, 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 2.78. Events Generated

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.1.19 cmd_gatt_write_characteristic_value_without_response

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

Table 2.79. 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 This value is normally received from the gatt_characteristic event.
7	uint8array	value	Characteristic value

Table 2.80. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_va
lue_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;
};
```

2.5.1.20 cmd_gatt_write_descriptor_value

This command can be used to write the value of a characteristic descriptor in a remote GATT database. If the given value does not fit in one ATT PDU, "write long" GATT procedure is used automatically. Received gatt_procedure_completed event indicates that all data has been written succesfully or that an error response has been received.

Table 2.81. 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 2.82. 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	0x0f	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_write_descriptor_value_rsp_t *gecko_cmd_gatt_write_descriptor_value(uint8 connection, uin
t16 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 2.83. Events Generated

Event	Description
gatt_procedure_completed	Procedure has been successfully completed or failed with error.

2.5.2 gatt events

2.5.2.1 evt_gatt_characteristic

This event 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 2.84. 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

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;
};
```

2.5.2.2 evt_gatt_characteristic_value

This event indicates that the value of a characteristic in the remote GATT server was received. This event is triggered as a result of 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 reveals 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 2.85. 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 This value is normally received from the gatt_characteristic event.
7	uint8	att_opcode	Attribute opcode which informs 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;
};
```

2.5.2.3 evt_gatt_descriptor

This event indicates that a GATT characteristic descriptor in the remote GATT database was discovered. This event is generated after issuing the gatt_discover_descriptors command.

Table 2.86. 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

C Functions

```
/* Event id */
gecko_evt_gatt_descriptor_id

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

2.5.2.4 evt_gatt_descriptor_value

This event 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 2.87. 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;
};
```

2.5.2.5 evt_gatt_mtu_exchanged

This event indicates that an ATT_MTU exchange procedure has been completed. Parameter mtu describes new MTU size. MTU size 23 is used before this event is received.

Table 2.88. 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;
};
```

2.5.2.6 evt_gatt_procedure_completed

This event indicates that the current GATT procedure has been completed successfully or that it has failed with an error. All GATT commands excluding gatt_write_characteristic_value_without_response and gatt_send_characteristic_confirmation will trigger this event, so the application must wait for this event before issuing another GATT command (excluding the two aforementioned exceptions).

Table 2.89. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.5.2.7 evt_gatt_service

This event indicates 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 2.90. 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

C Functions

```
/* Event id */
gecko_evt_gatt_service_id

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

2.5.3 gatt enumerations

2.5.3.1 enum_gatt_att_opcode

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

Table 2.91. 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

2.5.3.2 enum_gatt_client_config_flag

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

Table 2.92. Enumerations

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

2.5.3.3 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 2.93. Enumerations

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

2.6 Generic Attribute Profile Server (gatt_server)

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

2.6.1 gatt_server commands

2.6.1.1 cmd_gatt_server_find_attribute

This command can be used to find attributes of certain type from a local GATT database. Type is usually given as 16-bit or 128-bit UUID.

Table 2.94. 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 index
6	uint8array	type	Variable length byte array. The first byte defines the length of the data that follows, 0 - 255 bytes.

Table 2.95. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 typ
e_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;
};
```

2.6.1.2 cmd_gatt_server_read_attribute_type

This command can be used to read the type of an attribute from a local GATT database. The type is a UUID, usually 16 or 128 bits long.

Table 2.96. 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 2.97. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8array	type	Variable length byte array. The first byte defines the length of the data that follows, 0 - 255 bytes.

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_server_read_attribute_type_rsp_t *gecko_cmd_gatt_server_read_attribute_type(uint16 attrib
ute);

/* 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;
};
```

2.6.1.3 cmd_gatt_server_read_attribute_value

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

Table 2.98. 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 2.99. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8array	value	Variable length byte array. The first byte defines the length of the data that follows, 0 - 255 bytes.

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_server_read_attribute_value_rsp_t *gecko_cmd_gatt_server_read_attribute_value(uint16 attr
ibute, 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;
};
```

2.6.1.4 cmd_gatt_server_send_characteristic_notification

This command can be used to send notifications or indications to one or more remote GATT clients. At most ATT_MTU - 3 amount of data can be sent once.

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

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

Table 2.100. 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	Handle of the connection over which the notification or indication is sent. Values: • **0xff:** Sends notification or indication to all connected devices. • **Other:** Connection handle
5-6	uint16	characteristic	Characteristic handle
7	uint8array	value	Value to be notified or indicated

Table 2.101. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	sent_len	The length of data sent out if only one connected device is the receiver; otherwise 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;
};
```

2.6.1.5 cmd_gatt_server_send_user_read_response

This command must be used to send a response to a user_read_request event. The response needs to be sent within 30 second, 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 normal 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 once. Client will continue reading by sending new read request with increased offset value if it receives ATT_MTU - 1 amount of data.

Table 2.102. 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 This value is normally received from the gatt_characteristic event.
7	uint8	att_errorcode	Attribute protocol error code • **0:** No error • **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 2.103. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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(uint
8 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;
};
```

2.6.1.6 cmd_gatt_server_send_user_write_response

This command must be used to 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 2.104. 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 This value is normally received from the gatt_characteristic event.
7	uint8	att_errorcode	Attribute protocol error code • **0:** No error • **Non-zero:** See Bluetooth specification, Host volume, Attribute Protocol, Error Codes table.

Table 2.105. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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(uin
t8 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;
};
```

2.6.1.7 cmd_gatt_server_set_capabilities

This command can be used to set which capabilities should be enabled in the local GATT database. A service is visible to remote GATT clients if at least one of its capabilities has been 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 for how to declare capabilities in 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 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 2.106. 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 enable.
8-11	uint32	reserved	Value 0 should be used on this reserved field. None-zero values are reserved for future, do not use now.

Table 2.107. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_server_set_capabilities_rsp_t *gecko_cmd_gatt_server_set_capabilities(uint32 caps, uint3
2 reserved);

/* Response id */
gecko_rsp_gatt_server_set_capabilities_id

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

2.6.1.8 (deprecated) cmd_gatt_server_set_database

Deprecated. Use GATT capability feature for dynamic configuration of services and characteristics in the local GATT database. See gatt_server_set_capabilities command for the details.

This command can be used to set the local GATT database. The database should not be changed while this device is connected as peripheral since it may cause GATT attributes and data synchronization problems. If the database is changed during advertising mode, advertising packets will not be updated until the advertising is restarted.

Table 2.108. 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	0x07	method	Message ID
4-7	uint32	ptr	The pointer to the GATT database

Table 2.109. 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	0x07	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_server_set_database_rsp_t *gecko_cmd_gatt_server_set_database(uint32 ptr);

/* Response id */
gecko_rsp_gatt_server_set_database_id

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

2.6.1.9 cmd_gatt_server_write_attribute_value

This command can be used to 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 in case such characteristic has property of 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 2.110. 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 2.111. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_gatt_server_write_attribute_value_rsp_t *gecko_cmd_gatt_server_write_attribute_value(uint16 at
tribute, 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;
};
```

2.6.2 gatt_server events

2.6.2.1 evt_gatt_server_attribute_value

This event indicates that the value of an attribute in the local GATT database has been changed by a remote GATT client. Parameter att_opcode describes which GATT procedure was used to change the value.

Table 2.112. 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 which informs the procedure from which attribute 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;
};
```

2.6.2.2 evt_gatt_server_characteristic_status

This event indicates either that a local Client Characteristic Configuration descriptor has been 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 has been sent with the gatt_server_send_characteristic_notification command, otherwise further GATT transactions over this connection are disabled by the stack.

Table 2.113. 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 This value is normally received from the gatt_characteristic event.
7	uint8	status_flags	Describes whether Client Characteristic Configuration was changed or if 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;
};
```

2.6.2.3 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 2.114. 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;
};
```

2.6.2.4 evt_gatt_server_user_read_request

This event 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 XML firmware configuration file to have type="user". 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 this GATT connection is dropped by remote side.

Table 2.115. 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 This value is normally received from the gatt_characteristic event.
7	uint8	att_opcode	Attribute opcode which informs the procedure from which attribute 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;
};
```

2.6.2.5 evt_gatt_server_user_write_request

This event indicates that a remote GATT client is attempting to write a value of an attribute in to the local GATT database, where the attribute was defined in the GATT XML firmware configuration file to have type="user". 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 this GATT connection is dropped 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 2.116. 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 This value is normally received from the gatt_characteristic event.
7	uint8	att_opcode	Attribute opcode which informs the procedure from which attribute 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;
};
```

2.6.3 gatt_server enumerations

2.6.3.1 enum_gatt_server_characteristic_status_flag

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

Table 2.117. Enumerations

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

2.7 Hardware (hardware)

The commands and events in this class can be used to access and configure the system hardware and peripherals.

2.7.1 hardware commands

2.7.1.1 cmd_hardware_enable_dcdc

Deprecated. This command can be used to enable or disable DC/DC.

Table 2.118. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x01	lolen	Minimum payload length
2	0x0c	class	Message class: Hardware
3	0x0d	method	Message ID
4	uint8	enable	Set DC/DC as enabled or disabled. • **0:** disabled • **1:** enabled

Table 2.119. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x0c	class	Message class: Hardware
3	0x0d	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_hardware_enable_dcdc_rsp_t *gecko_cmd_hardware_enable_dcdc(uint8 enable);

/* Response id */
gecko_rsp_hardware_enable_dcdc_id

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

2.7.1.2 cmd_hardware_get_time

Deprecated. Get elapsed time since last reset of RTCC

Table 2.120. 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 2.121. 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;
};
```

2.7.1.3 cmd_hardware_set_lazy_soft_timer

This command can be used to start a software timer with some slack. Slack parameter allows stack to optimize wake ups and save power. Timer event is triggered between time and time + slack. See also description of hardware_set_soft_timer command.

Table 2.122. 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). 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. 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, is returned in timeout event
13	uint8	single_shot	Timer mode. Values: • **0:** false (timer is repeating) • **1:** true (timer runs only once)

Table 2.123. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, uint3
2 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 2.124. Events Generated

Event	Description
hardware_soft_timer	Sent after specified interval

2.7.1.4 cmd_hardware_set_soft_timer

This command can be used to start a software timer. Multiple concurrent timers can be running simultaneously. There are 256 unique timer IDs 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 2.125. 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	Interval between how often to send events, in hardware clock ticks (1 second is equal to 32768 ticks). 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 corre-sponds to about 18.2 hours. If time is 0, removes the scheduled timer with the same handle.
8	uint8	handle	Timer handle to use, is returned in timeout event
9	uint8	single_shot	Timer mode. Values: • **0:** false (timer is repeating) • **1:** true (timer runs only once)

Table 2.126. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, ui
nt8 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 2.127. Events Generated

Event	Description
hardware_soft_timer	Sent after specified interval

2.7.2 hardware events

2.7.2.1 evt_hardware_soft_timer

This event indicates that the soft timer has lapsed.

Table 2.128. 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;
};
```

2.8 Connection management (le_connection)

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

2.8.1 le_connection commands

2.8.1.1 cmd_le_connection_close

This command can be used to close a Bluetooth connection or cancel an ongoing connection establishment process. The parameter is a connection handle which is reported in le_connection_opened event or le_gap_open response.

Table 2.129. 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 2.130. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.131. Events Generated

Event	Description
le_connection_closed	This event indicates that a connection was closed.

2.8.1.2 cmd_le_connection_disable_slave_latency

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

Table 2.132. 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

Table 2.133. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct  gecko_msg_le_connection_disable_slave_latency_rsp_t  *gecko_cmd_le_connection_disable_slave_latency(uint
8 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;
};
```

2.8.1.3 cmd_le_connection_get_rssi

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

Table 2.134. 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 2.135. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.136. Events Generated

Event	Description
le_connection_rssi	Triggered when this command has completed.

2.8.1.4 cmd_le_connection_set_parameters

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

Table 2.137. 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. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • 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. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s
9-10	uint16	latency	Slave latency. This parameter defines how many connection inter-vals the slave can skip if it has no data to send • Range: 0x0000 to 0x01f4 Use 0x0000 for default value
11-12	uint16	timeout	Supervision timeout. The supervision timeout defines for how long the connection is maintained despite the devices being unable to communicate at the currently configured connection intervals. • Range: 0x000a to 0x0c80 • Time = Value x 10 ms • Time Range: 100 ms to 32 s • The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds It is recommended that the supervision timeout is set at a value which allows communication attempts over at least a few connec-tion intervals.

Table 2.138. 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
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_le_connection_set_parameters_rsp_t *gecko_cmd_le_connection_set_parameters(uint8 connection, u
int16 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 2.139. Events Generated

Event	Description
le_connection_parameters	This event is triggered after new connection parameters has been applied on the con-nection.

2.8.1.5 cmd_le_connection_set_phy

This command can be used to set preferred PHYs for connection. Preferred PHYs are connection specific. Event le_connection_phy_status is received when PHY update procedure has been completed. Other than preferred PHY can also be set if remote device does not accept any of the preferred PHYs.

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

Table 2.140. 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 bitfield and multiple PHYs can be preferred by setting multiple bits. • **0x01:** 1 Mbit PHY • **0x02:** 2 Mbit PHY • **0x04:** 125 kbit Coded PHY (S=8) • **0x08:** 500 kbit Coded PHY (S=2)

Table 2.141. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.142. Events Generated

Event	Description
le_connection_phy_status	This event indicates that PHY update procedure has been completed.

2.8.2 le_connection events

2.8.2.1 evt_le_connection_closed

This event indicates that a connection was closed.

Table 2.143. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.8.2.2 evt_le_connection_opened

This event indicates that a new connection was opened, whether the devices are already bonded, and what is the role of the Bluetooth device (Slave or Master). An open connection can be closed with the le_connection_close command by giving the connection handle ID obtained from this event.

Table 2.144. 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: • **0:** Slave • **1:** Master
12	uint8	connection	Handle for new connection
13	uint8	bonding	Bonding handle. Values: • **0xff:** No bonding • **Other:** Bonding handle
14	uint8	advertiser	The local advertising set this connection was opened to. Values: • **0xff:** Invalid value or not applicable, and this field should be ignored • **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;
};
```

2.8.2.3 evt_le_connection_parameters

This event is triggered whenever the connection parameters are changed and at any time a connection is established.

Table 2.145. 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
7-8	uint16	latency	Slave latency
9-10	uint16	timeout	Supervision timeout
11	uint8	security_mode	Connection security mode
12-13	uint16	txsize	Maximum Data Channel PDU Payload size 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;
};
```

2.8.2.4 evt_le_connection_phy_status

This event indicates that PHY update procedure has been completed.

Table 2.146. 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	
5	uint8	phy	Current active PHY. See values from le_connection_set_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;
};
```

2.8.2.5 evt_le_connection_rssi

This event is triggered when an le_connection_get_rssi command has completed.

Table 2.147. 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: • **0x00:** The command succeeded • **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. This parameter should be ignored if the command failed. Range: -127 to +20. Units: dBm.

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;
};
```

2.8.3 le_connection enumerations

2.8.3.1 enum_le_connection_security

These values indicate the Bluetooth Security Mode.

Table 2.148. 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

2.9 Generic Access Profile (le_gap)

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

2.9.1 le_gap commands

2.9.1.1 cmd_le_gap_bt5_set_adv_data

This command can be used together with le_gap_start_advertising to advertise user defined data. First use this command to set the data in advertising packets and/or in the scan response packets, and then use command le_gap_start_advertising to start the advertising in user_data mode.

The maximum data length is 31 bytes for legacy advertising and 191 bytes for extended advertising. If advertising mode is currently active the new advertising data will be used immediately.

The invalid parameter error will be returned in following situations:

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

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

Table 2.149. 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 index, number of sets available is defined in stack configuration
5	uint8	scan_rsp	This value selects if the data is intended for advertising packets, scan response packets or advertising packet in OTA. Values: • **0:** Advertising packets • **1:** Scan response packets • **2:** OTA advertising packets • **4:** OTA scan response packets
6	uint8array	adv_data	Data to be set. Maximum data length: • **31** bytes for legacy advertising; • **191** bytes for extended advertising

Table 2.150. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.1.2 (deprecated) cmd_le_gap_bt5_set_adv_parameters

Deprecated. Replacements are le_gap_set_advertise_timing command for setting the advertising intervals, le_gap_set_advertise_channel_map command for setting the channel map, and le_gap_set_advertise_report_scan_request command for enabling and disabling scan request notifications.

Table 2.151. 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 index, number of sets available is defined in stack configuration
5-6	uint16	interval_min	Minimum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s Default value: 100 ms
7-8	uint16	interval_max	Maximum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s • Note: interval_max should be bigger than interval_min Default value: 200 ms
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 bit-mask. Values: • **1:** Advertise on CH37 • **2:** Advertise on CH38 • **3:** Advertise on CH37 and CH38 • **4:** Advertise on CH39 • **5:** Advertise on CH37 and CH39 • **6:** Advertise on CH38 and CH39 • **7:** Advertise on all channels Recommended value: 7 Default value: 7
10	uint8	report_scan	If non-zero, enables scan request notification, and scan requests will be reported as events. Default value: 0

Table 2.152. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, uin
t16 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 2.153. Events Generated

Event	Description
le_gap_scan_request	Triggered when a scan request has been received during the advertising if scan request notification has been enabled by this command.

2.9.1.3 (deprecated) cmd_le_gap_bt5_set_mode

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

Table 2.154. 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 index, number of sets available is defined in stack configuration
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 advertiser. Value 0 indicates no maximum number limit.
9	uint8	address_type	Address type to use for packets

Table 2.155. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 c
onnect, 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 2.156. Events Generated

Event	Description
le_gap_adv_timeout	Triggered when the number of advertising events set by this command has been done and advertising is stopped on the given advertising set.
le_connection_opened	Triggered when a remote device opened a connection to the advertiser on the specified advertising set.

2.9.1.4 cmd_le_gap_clear_advertise_configuration

This command can be used to disable advertising configurations on the given advertising set. The command le_gap_set_advertise_configuration can be used to set configurations. This setting will take effect on the next advertising enabling.

Table 2.157. 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 index
5-8	uint32	configurations	Advertising configuration flags to disable. This value can be a bitmask of multiple flags. See le_gap_set_advertise_configuration for possible flags.

Table 2.158. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_le_gap_clear_advertise_configuration_rsp_t *gecko_cmd_le_gap_clear_advertise_configuration(uin
t8 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;
};
```

2.9.1.5 cmd_le_gap_connect

This command can be used to connect an advertising device with the specified initiating PHY. The Bluetooth stack will enter a state where it continuously scans for the connectable advertising packets from the remote device which matches the Bluetooth address given as a parameter. 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 the response from this command.

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

This command fails with "Connection Limit Exceeded" error if the number of connections attempted to be opened exceeds the max_connections value configured.

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

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

Table 2.159. 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: • **1:** LE 1M PHY • **4:** LE Coded PHY

Table 2.160. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8	connection	Handle that will be assigned to the connection once the connection will be 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 init
iating_phy);
```

```
/* Response id */
gecko_rsp_le_gap_connect_id

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

Table 2.161. Events Generated

Event	Description
le_connection_opened	This event is triggered after the connection has been opened, and indicates whether the devices are already bonded and what is the role of the Bluetooth device (Slave or Master).
le_connection_parameters	This event indicates the connection parameters and security mode of the connection.

2.9.1.6 (deprecated) cmd_le_gap_discover

Deprecated. Replacement is le_gap_start_discovery command which allows to scan on LE 1M PHY or LE Coded PHY.

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

Table 2.162. 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 2.163. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.164. Events Generated

Event	Description
le_gap_scan_response	Every 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.

2.9.1.7 cmd_le_gap_end_procedure

This command can be used to end a current GAP procedure.

Table 2.165. 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 2.166. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.1.8 (deprecated) cmd_le_gap_open

Deprecated. Replacement is le_gap_connect command which allows to open a connection with a specified PHY.

This command can be used to connect an advertising device with initiating PHY being the LE 1M PHY.

Table 2.167. 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	Address of the device to connect to
10	uint8	address_type	Address type of the device to connect to

Table 2.168. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8	connection	Handle that will be assigned to the connection once the connection will be 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_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 2.169. Events Generated

Event	Description
le_connection_opened	This event is triggered after the connection has been opened, and indicates whether the devices are already bonded and what is the role of the Bluetooth device (Slave or Master).
le_connection_parameters	This event indicates the connection parameters and security mode of the connection.

2.9.1.9 (deprecated) cmd_le_gap_set_adv_data

Deprecated. Use le_gap_bt5_set_adv_data command to set the 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 2.170. 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 the data is intended for advertising packets, scan response packets or advertising packet in OTA. Values: • **0:** Advertising packets • **1:** Scan response packets • **2:** OTA advertising packets • **4:** OTA scan response packets
5	uint8array	adv_data	Data to be set. Maximum data length: • **31** bytes for legacy advertising; • **191** bytes for extended advertising

Table 2.171. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, c
onst 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;
};
```

2.9.1.10 (deprecated) cmd_le_gap_set_adv_parameters

Deprecated. Replacements are le_gap_set_advertise_timing command for setting the advertising intervals, and le_gap_set_advertise_channel_map command for setting the channel map.

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

Table 2.172. 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 • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s Default value: 100 ms
6-7	uint16	interval_max	Minimum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s • Note: interval_max should be bigger than interval_min Default value: 200 ms
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 bit-mask. Values: • **1:** Advertise on CH37 • **2:** Advertise on CH38 • **3:** Advertise on CH37 and CH38 • **4:** Advertise on CH39 • **5:** Advertise on CH37 and CH39 • **6:** Advertise on CH38 and CH39 • **7:** Advertise on all channels Recommended value: 7 Default value: 7

Table 2.173. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, uint
16 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;
};
```

2.9.1.11 (deprecated) cmd_le_gap_set_adv_timeout

Deprecated. New command le_gap_set_advertise_timing should be used for this functionality.

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

Table 2.174. 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 2.175. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.1.12 cmd_le_gap_set_advertise_channel_map

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

Table 2.176. 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 index
5	uint8	channel_map	Advertisement channel map which determines which of the three channels will be used for advertising. This value is given as a bit-mask. Values: • **1:** Advertise on CH37 • **2:** Advertise on CH38 • **3:** Advertise on CH37 and CH38 • **4:** Advertise on CH39 • **5:** Advertise on CH37 and CH39 • **6:** Advertise on CH38 and CH39 • **7:** Advertise on all channels Recommended value: 7 Default value: 7

Table 2.177. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 handl
e, 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;
};
```

2.9.1.13 cmd_le_gap_set_advertise_configuration

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

Table 2.178. 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 index
5-8	uint32	configurations	Advertising configuration flags to enable. This value can be a bitmask of multiple flags. Flags: • **1 (Bit 0):** Use legacy advertising PDUs. • **2 (Bit 1):** Omit advertiser's address from all PDUs (anonymous advertising). This flag is effective only in extended advertising. • **4 (Bit 2):** Use le_gap_non_resolvable address type. Advertising must be in non-connectable mode if this configuration is enabled. • **8 (Bit 3):** Include TX power in advertising packets. This flag is effective only in extended advertising. Default value: 1

Table 2.179. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_le_gap_set_advertise_configuration_rsp_t *gecko_cmd_le_gap_set_advertise_configuration(uint8 h
andle, 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;
};
```

2.9.1.14 cmd_le_gap_set_advertise_phy

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

Table 2.180. 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 index
5	uint8	primary_phy	The PHY on which the advertising packets are transmitted on the primary advertising channel. If legacy advertising PDUs are being used, the PHY must be LE 1M. Values: • **1:** Advertising PHY is LE 1M • **4:** Advertising PHY is LE Coded Default: 1
6	uint8	secondary_phy	The PHY on which the advertising packets are transmitted on the secondary advertising channel. Values: • **1:** Advertising PHY is LE 1M • **2:** Advertising PHY is LE 2M • **4:** Advertising PHY is LE Coded Default: 1

Table 2.181. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.1.15 cmd_le_gap_set_advertise_report_scan_request

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

Table 2.182. 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 index
5	uint8	report_scan_req	If non-zero, enables scan request notification, and scan requests will be reported as events. Default value: 0

Table 2.183. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_req
uest(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 2.184. Events Generated

Event	Description
le_gap_scan_request	Triggered when a scan request has been received during the advertising if scan request notification has been enabled by this command.

2.9.1.16 cmd_le_gap_set_advertise_timing

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

Table 2.185. 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 index
5-8	uint32	interval_min	Minimum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s Default value: 100 ms
9-12	uint32	interval_max	Maximum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s • Note: interval_max should be bigger than interval_min Default value: 200 ms
13-14	uint16	duration	The advertising duration for this advertising set. Value 0 indicates no advertising duration limit and the advertising continues until it is disabled. A non-zero value sets the duration in units of 10 ms. The duration begins at the start of the first advertising event of this advertising set. • Range: 0x0001 to 0xFFFF • Time range: 10 ms to 655.35 s Default value: 0
15	uint8	maxevents	If non-zero, indicates the maximum number of advertising events to send before stopping advertiser. Value 0 indicates no maximum number limit. Default value: 0

Table 2.186. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.1.17 cmd_le_gap_set_advertise_tx_power

This command can be used to set the maximum power level at which the advertising packets can be transmitted. The RF path gain configuration and power output capability of the device will affect the actual output power, thus the selected output power level may be different than the given value. The maximum TX power of legacy advertising is always clipped to 10 dBm. Extended advertising TX power can be higher than 10 dBm if Adaptive Frequency Hopping has been enabled.

This setting will take effect on the next advertising enabling.

Table 2.187. 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	0x1b	method	Message ID
4	uint8	handle	Advertising set handle index
5	int8	power	The maximum power level Units: 1 dBm. Value range: -127 to 126. Value 127 indicates the application has no preference on this setting. Default: 127

Table 2.188. 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	0x1b	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

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, int
8 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;
};
```

2.9.1.18 cmd_le_gap_set_conn_parameters

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

Table 2.189. 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 be less than or equal to max_interval. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s Default value: 125 ms
6-7	uint16	max_interval	Maximum value for the connection event interval. This must be set greater than or equal to min_interval. • Time = Value x 1.25 ms • Range: 0x0006 to 0x0c80 • Time Range: 7.5 ms to 4 s Default value: 250 ms
8-9	uint16	latency	Slave latency. This parameter defines how many connection intervals the slave can skip if it has no data to send • Range: 0x0000 to 0x01f4 Default value: 0
10-11	uint16	timeout	Supervision timeout. The supervision timeout defines for how long the connection is maintained despite the devices being unable to communicate at the currently configured connection intervals. • Range: 0x000a to 0x0c80 • Time = Value x 10 ms • Time Range: 100 ms to 32 s • The value in milliseconds must be larger than (1 + latency) * max_interval * 2, where max_interval is given in milliseconds It is recommended that the supervision timeout is set at a value which allows communication attempts over at least a few connection intervals. Default value: 1000 ms

Table 2.190. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, ui
nt16 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;
};
```

2.9.1.19 cmd_le_gap_set_data_channel_classification

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

Table 2.191. 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. • **0:** Channel n is bad. • **1:** Channel n is unknown. The most significant bits are reserved and shall be set to 0 for future use. At least two channels shall be marked as unknown.

Table 2.192. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct  gecko_msg_le_gap_set_data_channel_classification_rsp_t  *gecko_cmd_le_gap_set_data_channel_classificatio
n(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;
};
```

2.9.1.20 cmd_le_gap_set_discovery_timing

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

Table 2.193. 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	The PHYs for which the parameters are set. • **1:** LE 1M PHY • **4:** LE Coded PHY • **5:** LE 1M PHY and LE Coded PHY
5-6	uint16	scan_interval	Scan interval. This is defined as the time interval from when the device started its last scan until it begins the subsequent scan, that is how often to scan • Time = Value x 0.625 ms • Range: 0x0004 to 0xFFFF • Time Range: 2.5 ms to 40.96 s Default value: 10 ms There is a variable delay when switching channels at the end of each scanning interval which is included in the scanning interval time itself. During this switch time no advertising packets will be received by the device. The switch time variation is dependent on use case, for example in case of scanning while keeping active connections the channel switch time might be longer than when only scanning without any active connections. Increasing the scanning interval will reduce the amount of time in which the device cannot receive advertising packets as it will switch channels less often. After every scan interval the scanner will change the frequency it operates at. It will cycle through all the three advertising channels in a round robin fashion. According to the specification all three channels must be used by a scanner.
7-8	uint16	scan_window	Scan window. The duration of the scan. scan_window shall be less than or equal to scan_interval • Time = Value x 0.625 ms • Range: 0x0004 to 0xFFFF • Time Range: 2.5 ms to 40.96 s Default value: 10 ms Note that packet reception is aborted if it has been started before scan window ends.

Table 2.194. 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

Byte	Type	Name	Description
3	0x16	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 sc
an_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;
};
```

2.9.1.21 cmd_le_gap_set_discovery_type

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

Table 2.195. 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	The PHYs for which the parameters are set. • **1:** LE 1M PHY • **4:** LE Coded PHY • **5:** LE 1M PHY and LE Coded PHY
5	uint8	scan_type	Scan type indicated by a flag. Values: • **0:** Passive scanning • **1:** Active scanning • In passive scanning mode the device only listens to advertising packets and will not transmit any packet • In active scanning mode the device will send out a scan request packet upon receiving advertising packet from a remote device and then it will listen to the scan response packet from remote device Default value: 0

Table 2.196. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_ty
pe);

/* Response id */
gecko_rsp_le_gap_set_discovery_type_id

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

2.9.1.22 (deprecated) cmd_le_gap_set_mode

Deprecated. Use le_gap_start_advertising command for enabling the advertising, and le_gap_stop_advertising command for disabling the advertising.

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

Table 2.197. 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 2.198. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.199. Events Generated

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

2.9.1.23 cmd_le_gap_set_privacy_mode

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

By default, privacy feature is disabled.

Table 2.200. 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: • **0:** Disable privacy • **1:** Enable privacy
5	uint8	interval	The minimum time interval between private address change. This parameter is ignored if this command is issued for disabling privacy mode. Values: • **0:** Use default interval, 15 minutes • **others:** The time interval in minutes

Table 2.201. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.1.24 (deprecated) cmd_le_gap_set_scan_parameters

Deprecated. Replacements are le_gap_set_discovery_timing command for setting timing parameters, and le_gap_set_discovery_type command for the scan type.

The parameters set by this command is only effective on the LE 1M PHY. For LE Coded PHY, above replacement command must be used.

Table 2.202. 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. This is defined as the time interval from when the device started its last scan until it begins the subsequent scan, that is how often to scan • Time = Value x 0.625 ms • Range: 0x0004 to 0x4000 • Time Range: 2.5 ms to 10.24 s Default value: 10 ms There is a variable delay when switching channels at the end of each scanning interval which is included in the scanning interval time itself. During this switch time no advertising packets will be received by the device. The switch time variation is dependent on use case, for example in case of scanning while keeping active connections the channel switch time might be longer than when only scanning without any active connections. Increasing the scanning interval will reduce the amount of time in which the device cannot receive advertising packets as it will switch channels less often. After every scan interval the scanner will change the frequency it operates at. It will cycle through all the three advertising channels in a round robin fashion. According to the specification all three channels must be used by a scanner.
6-7	uint16	scan_window	Scan window. The duration of the scan. scan_window shall be less than or equal to scan_interval • Time = Value x 0.625 ms • Range: 0x0004 to 0x4000 • Time Range: 2.5 ms to 10.24 s Default value: 10 ms Note that packet reception is aborted if it has been started before scan window ends.
8	uint8	active	Scan type indicated by a flag. Values: • **0:** Passive scanning • **1:** Active scanning • In passive scanning mode the device only listens to advertising packets and will not transmit any packet • In active scanning mode the device will send out a scan request packet upon receiving advertising packet from a remote device and then it will listen to the scan response packet from remote device Default value: 0

Table 2.203. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, u
int16 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;
};
```

2.9.1.25 cmd_le_gap_start_advertising

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

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

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

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

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

If advertising will be enabled in user_data mode, le_gap_bt5_set_adv_data should be used to set advertising and scan response data before issuing this command. When the advertising is enabled in other modes than user_data, the advertising and scan response data is generated by the stack using the following procedure:

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

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

Table 2.204. 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 index
5	uint8	discover	Discoverable mode
6	uint8	connect	Connectable mode

Table 2.205. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length

Byte	Type	Name	Description
2	0x03	class	Message class: Generic Access Profile
3	0x14	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_le_gap_start_advertising_rsp_t *gecko_cmd_le_gap_start_advertising(uint8 handle, uint8 discove
r, 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 2.206. Events Generated

Event	Description
le_gap_adv_timeout	Triggered when the number of advertising events set by this command has been done and advertising is stopped on the given advertising set.
le_connection_opened	Triggered when a remote device opened a connection to the advertiser on the specified advertising set.

2.9.1.26 cmd_le_gap_start_discovery

This command can be used to start the GAP discovery procedure to scan for advertising devices on the specified scanning PHY, that is to perform a device discovery. To cancel an ongoing discovery process use the le_gap_end_procedure command.

"Invalid Parameter" error will be returned if the scanning PHY value is invalid or the device does not support the PHY.

Table 2.207. 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: • **1:** LE 1M PHY • **4:** LE Coded PHY
5	uint8	mode	Bluetooth discovery Mode. For values see link

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	0x18	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.209. Events Generated

Event	Description
le_gap_scan_response	Every 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.

2.9.1.27 cmd_le_gap_stop_advertising

This command can be used to stop the advertising of the given advertising set.

Table 2.210. 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 index

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	0x15	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.9.2 le_gap events

2.9.2.1 evt_le_gap_adv_timeout

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

Table 2.212. 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;
};
```

2.9.2.2 evt_le_gap_scan_request

This event reports any scan request received in advertising mode if scan request notification is enabled

Table 2.213. 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	The 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: • **0:** Public address • **1:** Random address
12	uint8	bonding	Bonding handle if the remote scanning device has previously bon-ded with the local device. Values: • **0xff:** No bonding • **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;
};
```

2.9.2.3 evt_le_gap_scan_response

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

Table 2.214. 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 • Range: -127 to +20 • Units: dBm
5	uint8	packet_type	**Bits 0..2**: advertising packet type • **000**: Connectable scannable undirected advertising • **001**: Connectable undirected advertising • **010**: Scannable undirected advertising • **011**: Non-connectable non-scannable undirected advertising • **100**: Scan Response. Note: this is received only if the device is in active scan mode. **Bits 3..4**: Reserved for future **Bits 5..6**: data completeness • **00:** Complete • **10:** Incomplete, data truncated, no more to come **Bit 7**: legacy or extended advertising • **0:** Legacy advertising PDUs used • **1:** Extended advertising PDUs used
6-11	bd_addr	address	Bluetooth address of the remote device
12	uint8	address_type	Advertiser address type. Values: • **0:** Public address • **1:** Random address • **255:** No address provided (anonymous advertising)
13	uint8	bonding	Bonding handle if the remote advertising device has previously bonded with the local device. Values: • **0xff:** No bonding • **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;
};
```

2.9.3 le_gap enumerations

2.9.3.1 enum_le_gap_address_type

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

Table 2.215. Enumerations

Value	Name	Description
0	le_gap_address_type_public	Public address
1	le_gap_address_type_random	Random address
2	le_gap_address_type_public_identity	Public identity address resolved by stack
3	le_gap_address_type_random_identity	Random identity address resolved by stack

2.9.3.2 enum_le_gap_adv_address_type

Address type to use for advertising

Table 2.216. Enumerations

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

2.9.3.3 enum_le_gap_connectable_mode

These values define the available connectable modes.

Table 2.217. 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. **Deprecated**, replaced by enum le_gap_connectable_scannable. 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.

2.9.3.4 enum_le_gap_discover_mode

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

Table 2.218. 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

2.9.3.5 enum_le_gap_discoverable_mode

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

Table 2.219. 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 by using the Observation procedure
4	le_gap_user_data	Send advertising and/or scan response data defined by the user using le_gap_bt5_set_adv_data. The limited/ general discoverable flags are defined by the user.

2.9.3.6 enum_le_gap_phy_type

Types of PHYs used within le_gap class.

Table 2.220. Enumerations

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

2.10 Bluetooth Mesh Friend Node API (mesh_friend)

These commands and events are for Friend operation, available in nodes which have the Friend feature.

2.10.1 mesh_friend commands

2.10.1.1 cmd_mesh_friend_deinit

Deinitialize the Friend functionality. After calling this command, a possible friendship with a Low Power node is terminated and all friendsips are terminated. After calling this command, no other command in this class should be called before the Friend mode is initial-ized again.

Table 2.221. 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 2.222. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.10.1.2 cmd_mesh_friend_init

Initialize the Friend mode. The node needs to be provisioned before calling this command. Once 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 2.223. 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 2.224. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.10.2 mesh_friend events

2.10.2.1 evt_mesh_friend_friendship_established

Indication that a friendship has been established

Table 2.225. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x02	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

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;
};
```

2.10.2.2 evt_mesh_friend_friendship_terminated

Indication that a friendship that was successfully established has been terminated

Table 2.226. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x02	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

C Functions

```
/* Event id */
gecko_evt_mesh_friend_friendship_terminated_id

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

2.11 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; messages; it is up to the application to implement any more complex functionality (state machines or other model specific logic). The 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 how messages are formatted, transition time and remaining time resolution units depend on the requested or reported value: until 6.2 seconds it is 100ms; until 62 seconds it is 1s; until 620 seconds it is 10s; and until 620 minutes it is 10 minutes. The value cannot be longer than 620 minutes. Thus, for instance, it is not possible to request a delay of exactly 7500ms; the resolution unit is 1s between 6.2 and 62 seconds, so the value would be rounded down to 7s. Delay resolution is 5ms and values will be rounded down to the closest 5ms. The value cannot be longer than 1275ms.

2.11.1 mesh_generic_client commands

2.11.1.1 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 2.227. 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 2.228. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.229. 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.

2.11.1.2 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 2.230. 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 2.231. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.232. 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.

2.11.1.3 cmd_mesh_generic_client_init

Initialize generic client models

Table 2.233. 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 2.234. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.11.1.4 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 2.235. 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. 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. 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: • **Bit 0:** Response required. If nonzero client expects a response from the server
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 2.236. 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
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_generic_client_publish_rsp_t *gecko_cmd_mesh_generic_client_publish(uint16 model_id, uint
16 elem_index, uint8 tid, uint32 transition, uint16 delay, uint16 flags, uint8 type, uint8 parameters_len, cons
t 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 2.237. 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.

2.11.1.5 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 erver 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 2.238. 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. 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. 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: • **Bit 0:** Response required. If nonzero client expects a response from the server
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 2.239. 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
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, ui
nt8 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 2.240. 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.

2.11.2 mesh_generic_client events

2.11.2.1 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 2.241. 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 the message was sent to; 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: • **Bit 0:** Nonrelayed. If nonzero indicates a response to a nonrelayed 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;
};
```

2.11.3 mesh_generic_client defines

2.11.3.1 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 2.242. 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

2.11.3.2 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 2.243. 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

2.12 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. The 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 how messages are formatted, transition time and remaining time resolution units depend on the requested or reported value: until 6.2 seconds it is 100ms; until 62 seconds it is 1s; until 620 seconds it is 10s; and until 620 minutes it is 10 minutes. The value cannot be longer than 620 minutes. Thus, for instance, it is not possible to request a delay of exactly 7500ms; the resolution unit is 1s between 6.2 and 62 seconds, so the value would be rounded down to 7s. Delay resolution is 5ms and values will be rounded down to the closest 5ms. The value cannot be longer than 1275ms.

2.12.1 mesh_generic_server commands

2.12.1.1 cmd_mesh_generic_server_init

Initialize generic server models

Table 2.244. 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 2.245. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.12.1.2 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 2.246. 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 2.247. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_generic_server_publish_rsp_t *gecko_cmd_mesh_generic_server_publish(uint16 model_id, uint
16 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;
};
```

2.12.1.3 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 2.248. 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 transition time does not apply to the state change.
16-17	uint16	flags	Message flags. Bitmask of the following: • **Bit 0:** Nonrelayed. If nonzero indicates a response to a nonrelayed 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 2.249. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_generic_server_response_rsp_t *gecko_cmd_mesh_generic_server_response(uint16 model_id, ui
nt16 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;
};
```

2.12.1.4 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 2.250. 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 2.251. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_generic_server_update_rsp_t *gecko_cmd_mesh_generic_server_update(uint16 model_id, uint1
6 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;
};
```

2.12.2 mesh_generic_server events

2.12.2.1 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 2.252. 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; can be either the model element's unicast address, or a subscription address of the model
12-13	uint16	appkey_index	The application key index used in encrypting the request; any response needs 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. This applies to messages 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. This applies to messages 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: • **Bit 0:** Nonrelayed. If nonzero indicates that the client message was not relayed (TTL was zero); this means the server is within direct radio range of the client. • **Bit 1:** Response required. If nonzero 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;
};
```

2.12.2.2 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 2.253. 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;
};
```

2.13 Bluetooth Mesh Health Client Model (mesh_health_client)

Bluetooth Mesh Health client model functionality.

2.13.1 mesh_health_client commands

2.13.1.1 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 (in case the destination server address is a group address) from the network will generate server status report events.

Table 2.254. 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 nonzero a reliable model message is used.

Table 2.255. 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 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 s
erver_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;
};
```

2.13.1.2 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 (in case the destination server address is a group address) from the network will generate server status report events.

Table 2.256. 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 2.257. 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 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 serve
r_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;
};
```

2.13.1.3 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 (in case the destination server address is a group address) from the network will generate server status report events.

Table 2.258. 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 2.259. 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 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;
};
```

2.13.1.4 cmd_mesh_health_client_get_period

Get the health period log of a Health Server model or models in the network. Besides the immediate result code the response or responses (in case the destination server address is a group address) from the network will generate server status report events.

Table 2.260. 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. 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 2.261. 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 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;
};
```

2.13.1.5 cmd_mesh_health_client_set_attention

Set the attention timer value of a Health Server model or models in the network. Besides the immediate result code the response or responses (in case the destination server address is a group address) from the network will generate server status report events.

Table 2.262. 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 nonzero a reliable model message is used.

Table 2.263. 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 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;
};
```

2.13.1.6 cmd_mesh_health_client_set_period

Set the health period divisor of a Health Server model or models in the network. Besides the immediate result code the response or responses (in case the destination server address is a group address) from the network will generate server status report events.

Table 2.264. 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. 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 nonzero a reliable model message is used.

Table 2.265. 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 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;
};
```

2.13.1.7 cmd_mesh_health_client_test

Execute a self test on a server model or models in the network

Table 2.266. 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 nonzero a reliable model message is used.

Table 2.267. 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 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 ser
ver_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;
};
```

2.13.2 mesh_health_client events

2.13.2.1 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 2.268. 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. In case of an error (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 re-ceived 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	Unsigned 8-bit integer
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;
};
```

2.13.2.2 evt_mesh_health_client_server_status_attention

Receiving a Health Server attention status message generates this event.

Table 2.269. 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. In case of an error (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 re-ceived 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	Unsigned 8-bit integer

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;
};
```

2.13.2.3 evt_mesh_health_client_server_status_period

Receiving a Health Server period status message generates this event.

Table 2.270. 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. In case of an error (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	period	Unsigned 8-bit integer

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;
};
```

2.14 Bluetooth Mesh Health Server Model (mesh_health_server)

Bluetooth Mesh Health server model functionality.

2.14.1 mesh_health_server commands

2.14.1.1 cmd_mesh_health_server_clear_fault

Clear fault condition on an element.

Table 2.271. 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	Unsigned 16-bit integer
6	uint8	id	Unsigned 8-bit integer

Table 2.272. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_ind
ex, 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;
};
```

2.14.1.2 cmd_mesh_health_server_set_fault

Set fault condition on an element.

Table 2.273. 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	Unsigned 16-bit integer
6	uint8	id	Unsigned 8-bit integer

Table 2.274. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.14.1.3 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 2.275. 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	Unsigned 16-bit integer

Table 2.276. 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 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;
};
```

2.14.2 mesh_health_server events

2.14.2.1 evt_mesh_health_server_attention

. Attention timer on an element is set to a given value. This may happen, for instance, during provisioning. Application should use suitable means to get the user's attention, e.g., by vibrating or blinking a LED.

Table 2.277. 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	Unsigned 8-bit integer

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;
};
```

2.14.2.2 evt_mesh_health_server_test_request

. Health client request for a selftest generates this event. After the test has been executed, test results may need to be reported.

Table 2.278. 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; 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	Unsigned 8-bit integer
13-14	uint16	vendor_id	Unsigned 16-bit integer
15	uint8	response_required	Nonzero if client expects a response; 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;
};
```

2.15 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.

2.15.1 mesh_lpn commands

2.15.1.1 cmd_mesh_lpn_configure

Configure the parameters for friendship establishment

Table 2.279. 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. This value should be chosen based on the expected message frequency and LPN sleep period, as 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. Poll timeout is the longest time LPN will sleep 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 100ms Range: 1s to 95h 59 min 59s 900 ms

Table 2.280. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.15.1.2 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, no other command in this class should be called before the Low Power mode is initialized again.

Table 2.281. 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 2.282. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.15.1.3 cmd_mesh_lpn_establish_friendship

Establish a friendship. Once a frienship has been established the node can start saving power.

Table 2.283. 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 2.284. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_lpn_establish_friendship_rsp_t *gecko_cmd_mesh_lpn_establish_friendship(uint16 netkey_ind
ex);

/* Response id */
gecko_rsp_mesh_lpn_establish_friendship_id

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

2.15.1.4 cmd_mesh_lpn_init

Initialize the Low Power node (LPN) mode. The node needs to be provisioned before calling this command. Once the LPN mode is initialized, the node cannot operate in the network without a Friend node. In order to establish a friendship with a nearby Friend node, the establish friendship command should be used. This call has to be made before calling the other commands in this class.

Table 2.285. 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 2.286. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.15.1.5 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, as the procedure will be performed autonomously before the poll timeout expires.

Table 2.287. 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 2.288. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.15.1.6 cmd_mesh_lpn_terminate_friendship

Terminate an already established friendship. Friendship terminated event will be emitted when the friendship termination has been completed.

Table 2.289. 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 2.290. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the

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;
};
```

2.15.2 mesh_lpn events

2.15.2.1 evt_mesh_lpn_friendship_established

Indication that a friendship has been established

Table 2.291. 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;
};
```

2.15.2.2 evt_mesh_lpn_friendship_failed

Indication that friendship establishment failed

Table 2.292. 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;
};
```

2.15.2.3 evt_mesh_lpn_friendship_terminated

Indication that a friendship that was successfully established has been terminated

Table 2.293. 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;
};
```

2.16 Mesh Node (mesh_node)

Bluetooth Mesh API for unprovisioned devices and provisioned nodes. **Initialization:**

- Initialize node
-
- Node initialized

Provisioning a Node:

- Get device UUID
- Start 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

Note on 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 some address types. The address types are as follows:

- **0x0000 Unassigned address:** represents an address that has not been set
- **0x0001..0x7fff Unicast addresses:** addresses allocated by the Provisioner to provisioned nodes. Each element of a node has its own unicast address.
- **0x8000..0xbfff Virtual addresses:** 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 usually be used in the same contexts as group addresses. Some commands require specifying the full Label UUID instead of the virtual address shorthand.
- **0xc000..0xffef Group addresses:** Addresses allocated by the Provisioner for multicast communication.
- **0xfff0..0xffff Fixed group addresses:** Addresses allocated in the Mesh specification for multicast communication in a particular context. Can be used in the same contexts 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

2.16.1 mesh_node commands

2.16.1.1 cmd_mesh_node_clear_statistics

Table 2.294. 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 2.295. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.2 cmd_mesh_node_get_element_address

Get the unicast address configured to an element.

Table 2.296. 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 2.297. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	address	The address of the element. Returns 0x0000 if the address is not configured or in case of an error.

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;
};
```

2.16.1.3 cmd_mesh_node_get_ivrecovery_mode

Table 2.298. 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 2.299. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8	mode	Unsigned 8-bit integer

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;
};
```

2.16.1.4 cmd_mesh_node_get_ivupdate_state

Get the current IV index update state in the network.

Table 2.300. 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 2.301. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-9	uint32	ivindex	Current IV index
10	uint8	state	Whether 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;
};
```

2.16.1.5 cmd_mesh_node_get_net_relay_delay

Table 2.302. 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 2.303. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.6 cmd_mesh_node_get_seq_remaining

Get the number of sequence numbers remaining on an element (before sequence numbers are exhausted). Note that every element of a node keeps a separate sequence number counter. "

Table 2.304. 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 2.305. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.7 cmd_mesh_node_get_statistics

Table 2.306. 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 2.307. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.8 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, as 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 2.308. 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 2.309. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.9 cmd_mesh_node_init

Initializes the 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 2.310. 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 2.311. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.312. Events Generated

Event	Description
mesh_node_initialized	Node initialized and operational

2.16.1.10 cmd_mesh_node_init_oob

Initializes the Mesh stack in Node role. When initialization is complete a node initialized event will be 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 2.313. 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 nonzero, use the ECC key stored in persistent storage 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 allowed. Valid values are as follows: • **Bit 0:** No OOB is allowed • **Bit 1:** Static OOB is allowed • **Bit 2:** Input OOB is allowed • **Bit 3:** Output OOB is allowed
6-7	uint16	output_actions	Allowed OOB Output Action types
8	uint8	output_size	Maximum Output OOB size
9-10	uint16	input_actions	Allowed OOB Input Action types
11	uint8	input_size	Maximum Input OOB size
12-13	uint16	oob_location	Defines the OOB data location bitmask

Table 2.314. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, u
int16 output_actions, uint8 output_size, uint16 input_actions, uint8 input_size, uint16 oob_location);

/* Response id */
gecko_rsp_mesh_node_init_oob_id

/* Response structure */
```

```
struct gecko_msg_mesh_node_init_oob_rsp_t
{
  uint16 result;
};
```

Table 2.315. Events Generated

Event	Description
mesh_node_initialized	Node initialized and operational

2.16.1.11 cmd_mesh_node_input_oob_request_rsp

This command is used to provide the stack with the Input out-of-band authentication data which the Provisioner is displaying.

Table 2.316. 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 2.317. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_le
n, 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;
};
```

2.16.1.12 cmd_mesh_node_request_ivupdate

Table 2.318. 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 2.319. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.13 cmd_mesh_node_save_replay_protection_list

Save replay protection list to NVM

Table 2.320. 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 2.321. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.14 cmd_mesh_node_set_adv_event_filter

Table 2.322. 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 • 0x01: Connectable undirected advertising • 0x02: Scannable undirected advertising • 0x04: Non connectable undirected advertising • 0x08: Scan Response • 0x8000: Use gap data type. Don't use with other values
6	uint8array	gap_data_type	Event send when advertising packet contains the data type. Values defined in SIG Data Types Specification. Values must be set as two digit hex number, maximum 8 items.

Table 2.323. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, ui
nt8 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;
};
```

2.16.1.15 cmd_mesh_node_set_ivrecovery_mode

Table 2.324. 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; nonzero to enable

Table 2.325. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.16 cmd_mesh_node_set_net_relay_delay

Table 2.326. 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. Must be equal to or greather than the minimum.

Table 2.327. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.17 cmd_mesh_node_set_provisioning_data

Used to provision devices completely out-of-band. Provisioner's device database needs to be populated 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 2.328. 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	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 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 2.329. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 d
evice_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;
};
```

2.16.1.18 cmd_mesh_node_set_uuid

Write device UUID into persistent storage. This command must be called before the mesh stack is initialized; otherwise the change will not take effect before a reboot. Note that UUID must not be changed when the device is provisioned to a network. Furthermore, UUID should remain constant if a device has received a firmware update which requires reprovisioning of the device once the update has been applied (e.g., new elements are added by the update).

Table 2.330. 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	0x11	method	Message ID
4-19	uuid_128	uuid	UUID to set

Table 2.331. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.1.19 cmd_mesh_node_start_unprov_beaconing

Start sending Unprovisioned Device Beacons. This command makes an unprovisioned device available for provisioning. The device will start to send 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). In case of 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 generates; if provisioning fails with an error, a provisioning failed event will be generated instead. Once provisioned, the node elements have been allocated addresses and a network key has been deployed to the node, making the node ready for further configuration by the Provisioner. Note that the node is not yet fully ready for communicating with other nodes on the network at this stage.

Table 2.332. 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: • **1 (bit 0):** PB-ADV • **2 (bit 1):** PB-GATT Other bits are reserved and must not be used.

Table 2.333. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_node_start_unprov_beaconing_rsp_t *gecko_cmd_mesh_node_start_unprov_beaconing(uint8 beare
r);

/* 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 2.334. 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. The node is now ready for further configuration by the Provisioner, but it 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.

2.16.1.20 cmd_mesh_node_static_oob_request_rsp

This command is used to provide the stack with static out-of-band authentication data which the stack requested.

Table 2.335. 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 2.336. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.2 mesh_node events

2.16.2.1 evt_mesh_node_changed_ivupdate_state

Network IV index update state has changed

Table 2.337. 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	Whether 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;
};
```

2.16.2.2 evt_mesh_node_config_get

Informative; Configuration Client requested the current value of a State in the Configuration Server Model.

Table 2.338. 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;
};
```

2.16.2.3 evt_mesh_node_config_set

Informative; Configuration Client changes the State in the Configuration Server Model.

Table 2.339. 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;
};
```

2.16.2.4 evt_mesh_node_display_output_oob

Display Output OOB Data so Provisioner can input it.

Table 2.340. 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. Values are as follows: • **0x00:** Blink • **0x01:** Beep • **0x02:** Vibrate • **0x03:** Output Numeric • **0x04:** Output Alphanumeric
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;
};
```

2.16.2.5 evt_mesh_node_initialized

Node initialized and operational

Table 2.341. 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; this should be ignored if unprovisioned. Secondary elements have been assigned sequential unicast addressed following the primary element address.
7-10	uint32	ivi	IV index for the first network of the node; this should be ignored 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;
};
```

2.16.2.6 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 Mesh stack using the respond to input OOB request command.

Table 2.342. 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. Values are as follows: • **0x00:** Push • **0x01:** Twist • **0x02:** Input Numeric • **0x03:** Input Alphanumeric
5	uint8	input_size	Size of data in 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;
};
```

2.16.2.7 evt_mesh_node_ivrecovery_needed

Network IV index recovery is 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 2.343. 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	Unsigned 32-bit integer
8-11	uint32	network_ivindex	Unsigned 32-bit integer

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;
};
```

2.16.2.8 evt_mesh_node_key_added

This event is received when a Configuration Client has deployed a new network or application key to the node.

Table 2.344. 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: • **0x00:** Network key • **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; 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;
};
```

2.16.2.9 evt_mesh_node_model_config_changed

Informative. This event notifies that a remote Configuration Client has changed the configuration of a local model.

Table 2.345. 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_con-fig_state	The configuration state which has changed. Values are as follows: • **0x00:** Model application key bindings • **0x01:** Model publication parameters • **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;
};
```

2.16.2.10 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 it 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 2.346. 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 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;
};
```

2.16.2.11 evt_mesh_node_provisioning_failed

Provisioning the Node has failed.

Table 2.347. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.2.12 evt_mesh_node_provisioning_started

Provisioner has started provisioning this Node

Table 2.348. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.16.2.13 evt_mesh_node_reset

Provisioner has instructed the node to reset itself. 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 2.349. 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
{
};
```

2.16.2.14 evt_mesh_node_static_oob_request

Static out of band authentication data is needed in the provisioning. Application on the Node should provide the value to the Mesh stack using the respond to static OOB authentication data request command.

Table 2.350. 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
{
};
```

2.16.3 mesh_node enumerations

2.16.3.1 enum_mesh_node_config_state

Specifies the State to which a Configuration Client/Server command/event applies

Table 2.351. 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

2.17 Bluetooth Mesh Provisioner (mesh_prov)

Bluetooth Mesh API for embedded Provisioner. **Initialization:**

- Initialize provisioner
- Provisioner initialized

Provisioning a Node:

- Scan 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 and Mesh Network Management

- Create a new network key on the Provisioner
- Deploy a network key to a Node
- Remove a network key from a Node
- Create a new application key on the Provisioner
- Get list of application keys bound to a network key from a Node
- Deploy an application key to a Node
- Remove an application key from a Node
- Start a key refresh procedure
- Get node key refresh blacklist status
- Set node key refresh blacklist status

Node Configuration

- Get device composition data of a Node
- Device composition data of a Node received
- Get a Node configuration state value
- Set a Node configuration state value
- Node configuration state response
- Get node network retransmission configuration
- Set node network retransmission configuration
- Get node relay retransmission configuration
- Set node relay retransmission configuration
- Node relay retransmission status report

Model Configuration

- Bind a model to an application key
- Remove a model to application key binding
- Get the model to application key bindings of 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
-
- Set a model's publication parameters

Heartbeat

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

- Heartbeat publication status report from a node
- Get node heartbeat subscription settings
- Set node heartbeat subscription settings
- Heartbeat subscription status report from a node

Device Database

- Add a node to Device Database
- Remove a node from Device Database
- Fetch node data from Device Database
- Request a list of nodes in Device Database
- Device database list result

These commands are available only if Provisioner functionality has been compiled in the device. If that is not the case, a "feature not implemented" error code will be returned for all functions in this class.

2.17.1 mesh_prov commands

2.17.1.1 cmd_mesh_prov_appkey_add

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; that is, 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 2.352. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 Nnde.

Table 2.353. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0e	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.354. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.2 cmd_mesh_prov_appkey_delete

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 2.355. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 kkey is bound on the node
8-9	uint16	appkey_index	Index of the application key to delete

Table 2.356. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0f	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.357. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.3 cmd_mesh_prov_appkey_get

Get a list of application keys bound to a network key on a node. This command is used to 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 2.358. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.359. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2a	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.360. Events Generated

Event	Description
mesh_prov_appkey_list	Application key list event.
mesh_prov_appkey_list_end	Application key list terminator event.

2.17.1.4 cmd_mesh_prov_create_appkey

Creates a new application key on the Provisioner. An application key is always bound to a network key; that is, 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 2.361. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.362. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x05	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x07	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	appkey_index	Index of new Application Key; should be ignored if result was non-zero.
8	uint8array	key	New Application Key; should be ignored if result was nonzero.

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_create_appkey_rsp_t *gecko_cmd_mesh_prov_create_appkey(uint16 netkey_index, uint8 ke
y_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;
};
```

2.17.1.5 cmd_mesh_prov_create_network

Creates a new network key on the Provisioner. The created key can be deployed on a Node using the add network key command.

Table 2.363. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x01	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x03	method	Message ID
4	uint8array	key	Key value to use; set to zero-length array to generate random key.

Table 2.364. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x03	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.1.6 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 in order for it to be configurable. See set node provisioning data command.

Table 2.365. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x00	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 shall use 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 2.366. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x18	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, ui
nt16 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;
};
```

2.17.1.7 cmd_mesh_prov_ddb_delete

Delete node information from 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 2.367. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x00	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x17	method	Message ID
4-19	uuid_128	uuid	UUID of the node to delete

Table 2.368. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x17	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.1.8 cmd_mesh_prov_ddb_get

Get a Provisioner device database entry with matching UUID.

Table 2.369. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x01	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x16	method	Message ID
4	uint8array	uuid	UUID of the Device to retrieve

Table 2.370. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x16	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 provi-sioned. Used for network level encryption of Config Client mes-sages.
24-25	uint16	address	Unicast address of the primary element of the node
26	uint8	elements	Number of elements the node has

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;
};
```

2.17.1.9 cmd_mesh_prov_ddb_list_devices

Lists nodes known by this Provisioner. A number of database listing events will be generated.

Table 2.371. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x00	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x19	method	Message ID

Table 2.372. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x19	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.373. Events Generated

Event	Description
mesh_prov_ddb_list	Provisioner's device database list entry

2.17.1.10 cmd_mesh_prov_friend_timeout_get

friend poll timeout status event.

Table 2.374. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.375. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x32	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, uin
t16 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 2.376. Events Generated

Event	Description
mesh_prov_friend_timeout_status	Node relay state report.

2.17.1.11 cmd_mesh_prov_get_config

Get a configration state value of a Node. Node Configuration Server model state contains a number of node-wide values (for instance, Node's default TTL value) which are represented as single bytes; they can be queried with this command. See the list of configuration states for reference. Querying the more complex states (for instance, model-application key bindings) should be done 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 2.377. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.378. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x05	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 n
etkey_index);

/* Response id */
gecko_rsp_mesh_prov_get_config_id

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

Table 2.379. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.12 cmd_mesh_prov_get_dcd

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 2.380. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.381. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x04	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.1.13 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 thus be shut out of the network.

Table 2.382. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0c	method	Message ID
4-5	uint16	key	Network key index
6	uint8array	uuid	UUID of the Device

Table 2.383. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0c	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8	status	Nonzero 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(uint1
6 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;
};
```

2.17.1.14 cmd_mesh_prov_heartbeat_publication_get

Get heartbeat publication state of a node. Node response will be reported as a heartbeat publication status event.

Table 2.384. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.385. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x23	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_heartbeat_publication_get_rsp_t *gecko_cmd_mesh_prov_heartbeat_publication_get(uint1
6 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 2.386. Events Generated

Event	Description
mesh_prov_heartbeat_publication_status	Node heartbeat status, generated in response to a get heartbeat publication state or set heartbeat publication state request.

2.17.1.15 cmd_mesh_prov_heartbeat_publication_set

Set heartbeat publication state of a node. Node response will be reported as a heartbeat publication status event.

Table 2.387. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0d	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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: • **0x00:** Heartbeat messages are not sent • **0x01 .. 0x11:** Node shall send 2^(n-1) heartbeat messages • **0x12 .. 0xfe:** Prohibited • **0xff:** Hearbeat messages are sent indefinitely
11	uint8	period_log	Heartbeat publication period setting. Valid values are as follows: • **0x00:** Heartbeat messages are not sent • **0x01 .. 0x11:** Node shall send a heartbeat message every 2^(n-1) seconds • **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: • **Bit 0:** Relay feature • **Bit 1:** Proxy feature • **Bit 2:** Friend feature • **Bit 3:** Low power feature Remaining bits are reserved for future use.
15-16	uint16	publication_netkey_index	Index of the network key used to encrypt heartbeat messages.

Table 2.388. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x24	method	Message ID

Byte	Type	Name	Description
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_heartbeat_publication_set_rsp_t *gecko_cmd_mesh_prov_heartbeat_publication_set(uint1
6 address, uint16 netkey_index, uint16 publication_address, uint8 count_log, uint8 period_log, uint8 ttl, uint1
6 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 2.389. Events Generated

Event	Description
mesh_prov_heartbeat_publication_status	Node heartbeat status, generated in response to a get heartbeat publication state or set heartbeat publication state request.

2.17.1.16 cmd_mesh_prov_heartbeat_subscription_get

Get node heartbeat subscription state. The node will respond with a subscription status event.

Table 2.390. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.391. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x25	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_heartbeat_subscription_get_rsp_t *gecko_cmd_mesh_prov_heartbeat_subscription_get(uin
t16 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 2.392. Events Generated

Event	Description
mesh_prov_heartbeat_subscription_status	Node heartbeat subscription report.

2.17.1.17 cmd_mesh_prov_heartbeat_subscription_set

Get node heartbeat subscription state. The node will respond with a subscription status event.

Table 2.393. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x09	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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: • **0x00:** Heartbeat messages are not received • **0x01 .. 0x11:** Node shall receive heartbeat messages for 2^(n-1) seconds • **0x12 .. 0xff:** Prohibited

Table 2.394. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x26	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_heartbeat_subscription_set_rsp_t *gecko_cmd_mesh_prov_heartbeat_subscription_set(uin
t16 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 2.395. Events Generated

Event	Description
mesh_prov_heartbeat_subscription_status	Node heartbeat subscription report.

2.17.1.18 cmd_mesh_prov_init

Initializes the 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 commands. Note that you may initialize a device either in the Provisioner or the Node role, but not both.

Table 2.396. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x00	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x00	method	Message ID

Table 2.397. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x00	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.398. Events Generated

Event	Description
mesh_prov_initialized	Provisioner initialized and operational.

2.17.1.19 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. Thus the blacklisted nodes which did not receive new keys will be shut out of the network at the completion of the procedure.

Table 2.399. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 thus 2*num_appkeys bytes in total.

Table 2.400. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0b	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.401. 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.

Event	Description
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

2.17.1.20 cmd_mesh_prov_model_app_bind

Bind a model to an application key. Node response is reported with a configuration status event.

Table 2.402. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0c	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 model being configured
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 model being configured. Use 0xffff for Bluetooth SIG models.
14-15	uint16	model_id	Model ID of the model being configured.

Table 2.403. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x10	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.404. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.21 cmd_mesh_prov_model_app_get

Get application keys to which the model is bound. Node response is reported with a configuration status event.

Table 2.405. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0a	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 model being config-ured
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 model being configured.

Table 2.406. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x12	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.407. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.22 cmd_mesh_prov_model_app_unbind

Remove application key binding from a model. Node response is reported with a configuration status event.

Table 2.408. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0c	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 model being config-ured
8-9	uint16	netkey_index	The network key index used for encrypting the request.
10-11	uint16	appkey_index	The index of the application key used in the binding to be re-moved
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 model being configured.

Table 2.409. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x11	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.410. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.23 cmd_mesh_prov_model_pub_get

Get a model's publication address, key, and parameters. Node response is reported with a model publication parameters event.

Table 2.411. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x08	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 to be config-ured.
8-9	uint16	vendor_id	Vendor ID of model being configured. Use 0xffff for Bluetooth SIG models.
10-11	uint16	model_id	Model ID of the model being configured.

Table 2.412. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2d	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.413. Events Generated

Event	Description
mesh_prov_model_pub_status	

2.17.1.24 cmd_mesh_prov_model_pub_set

Set a model's publication address, key, and parameters. Node response is reported with a configuration status event.

Table 2.414. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x11	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 model to 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 model being configured. Use 0xffff for Bluetooth SIG models.
14-15	uint16	model_id	Model ID of the model being configured.
16-17	uint16	pub_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.
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: • **Bits 0..5:** Step count • **Bits 6..7:** Step resolution: • 00: 100 milliseconds • 01: 1 second • 10: 10 seconds • 11: 10 minutes
20	uint8	retrans	Retransmission count; controls how many times the model re-publishes the same message after the initial publish transmission. Range: 0..7. Default value is 0 (no retransmissions).

Table 2.415. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x14	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.416. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.25 cmd_mesh_prov_model_pub_set_cred

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 2.417. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x12	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 to 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 model being configured. Use 0xffff for Bluetooth SIG models.
14-15	uint16	model_id	Model ID of the model being configured.
16-17	uint16	pub_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.
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: • **Bits 0..5:** Step count • **Bits 6..7:** Step resolution: • 00: 100 milliseconds • 01: 1 second • 10: 10 seconds • 11: 10 minutes
20	uint8	retrans	Retransmission count; controls how many times the model re-publishes the same message after the initial publish transmission. Range: 0..7. Default value is 0 (no retransmissions).
21	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.

Table 2.418. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2f	method	Message ID

Byte	Type	Name	Description
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, uin
t16 elem_address, uint16 netkey_index, uint16 appkey_index, uint16 vendor_id, uint16 model_id, uint16 pub_addre
ss, 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 2.419. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.26 cmd_mesh_prov_model_pub_set_va

Set a model's publication virtual address, key, and parameters. Node response is reported with a configuration status event.

Table 2.420. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x10	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 to be config-ured.
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 model being configured.
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: • **Bits 0..5:** Step count • **Bits 6..7:** Step resolution: • 00: 100 milliseconds • 01: 1 second • 10: 10 seconds • 11: 10 minutes
18	uint8	retrans	Unsigned 8-bit integer
19	uint8array	pub_address	The Label UUID to publish to. The byte array must be exactly 16 bytes long.

Table 2.421. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2e	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 per
iod, 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 2.422. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.27 cmd_mesh_prov_model_pub_set_va_cred

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 2.423. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x11	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 to be config- ured.
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 model being configured.
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: • **Bits 0..5:** Step count • **Bits 6..7:** Step resolution: • 00: 100 milliseconds • 01: 1 second • 10: 10 seconds • 11: 10 minutes
18	uint8	retrans	Unsigned 8-bit integer
19	uint8	credentials	Friendship credential flag. If zero, publication is done using nor- mal 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.
20	uint8array	pub_address	The Label UUID to publish to. The byte array must be exactly 16 bytes long.

Table 2.424. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x30	method	Message ID

Byte	Type	Name	Description
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 addres
s, 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 2.425. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.28 cmd_mesh_prov_model_sub_add

Add an address to a model's subscription list. Node response is reported with a configuration status event.

Table 2.426. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0c	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 config- ured.
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 model being configured.
14-15	uint16	sub_address	The address to add to the subscription list. Note that the address has to be a group address.

Table 2.427. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x13	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.428. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.29 cmd_mesh_prov_model_sub_add_va

Add an virtual address to a model's subscription list. Node response is reported with a configuration status event.

Table 2.429. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0b	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 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 model being configured. Use 0xffff for Bluetooth SIG models.
12-13	uint16	model_id	Model ID of the model being configured.
14	uint8array	sub_address	The Label UUID to add to the subscription list. The array must be exactly 16 bytes long.

Table 2.430. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1f	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress_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 2.431. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.30 cmd_mesh_prov_model_sub_clear

Clear all addresses from a model's subscription list. Node response is reported with a configuration status event.

Table 2.432. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0a	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 to be config-ured.
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 model being configured.

Table 2.433. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2c	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 el
em_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 2.434. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.31 cmd_mesh_prov_model_sub_del

Remove an address from a model's subscription list. Node response is reported with a configuration status event.

Table 2.435. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0c	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 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 model being configured. Use 0xffff for Bluetooth SIG models.
12-13	uint16	model_id	Model ID of the model being configured.
14-15	uint16	sub_address	The address to remove from the subscription list

Table 2.436. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1e	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.437. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.32 cmd_mesh_prov_model_sub_del_va

Remove a virtual address from a Model's subscription list. Node response is reported with a configuration status event.

Table 2.438. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0b	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 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 model being configured. Use 0xffff for Bluetooth SIG models.
12-13	uint16	model_id	Model ID of the model being configured.
14	uint8array	sub_address	The Label UUID to remove from the subscription list. The array must be exactly 16 bytes long.

Table 2.439. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x20	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress_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 2.440. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.33 cmd_mesh_prov_model_sub_get

Get a model's subscription list. Node response is reported with subscription list entry and events.

Table 2.441. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0a	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 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 model being configured. Use 0xffff for Bluetooth SIG models.
12-13	uint16	model_id	Model ID of the model being configured.

Table 2.442. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x31	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.443. Events Generated

Event	Description
mesh_prov_model_sub_addr	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	This event terminates model subscription list result reporting.

2.17.1.34 cmd_mesh_prov_model_sub_set

Set an address to a model's subscription list, overwriting previous contents. Node response is reported with a configuration status event.

Table 2.444. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0c	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 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 model being configured. Use 0xffff for Bluetooth SIG models.
12-13	uint16	model_id	Model ID of the model being configured.
14-15	uint16	sub_address	The address to set as the subscription list. Note that the address has to be a group address.

Table 2.445. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x21	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress, 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 2.446. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.35 cmd_mesh_prov_model_sub_set_va

Set a virtual address to a model's subscription list, overwriting previous contents. Node response is reported with a configuration status event.

Table 2.447. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x0b	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 to be config- ured.
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 model being configured.
14	uint8array	sub_address	The Label UUID to set as the subscription list. The byte array must be exactly 16 bytes long.

Table 2.448. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x22	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_a
ddress_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 2.449. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.36 cmd_mesh_prov_nettx_get

Retrieve network layer transmission parameters of a node.

Table 2.450. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1c	method	Message ID
4-5	uint16	address	Unicast address of the target node

Table 2.451. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1c	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.1.37 cmd_mesh_prov_nettx_set

Set network layer transmission parameters of a node.

Table 2.452. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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	Rettransmission interval in 10-millisecond steps

Table 2.453. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1d	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 in
terval);

/* Response id */
gecko_rsp_mesh_prov_nettx_set_id

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

2.17.1.38 cmd_mesh_prov_network_add

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 2.454. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.455. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1a	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_ind
ex);

/* Response id */
gecko_rsp_mesh_prov_network_add_id

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

Table 2.456. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.39 cmd_mesh_prov_network_delete

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) as otherwise the node would not be able to respond. Node response is reported with an configuration status event.

Table 2.457. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.458. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x1b	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_network_delete_rsp_t *gecko_cmd_mesh_prov_network_delete(uint16 address, uint16 netk
ey_index);

/* Response id */
gecko_rsp_mesh_prov_network_delete_id

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

Table 2.459. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.40 cmd_mesh_prov_network_get

Get a list of network keys bound from a node. This command is used to 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 2.460. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2b	method	Message ID
4-5	uint16	address	Unicast address of the target Node's primary element

Table 2.461. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x2b	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.462. Events Generated

Event	Description
mesh_prov_network_list	Network key list event.
mesh_prov_network_list_end	Network key list terminator event.

2.17.1.41 cmd_mesh_prov_oob_auth_rsp

This command is used to respond to prov_oob_auth_request

Table 2.463. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x01	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x09	method	Message ID
4	uint8array	data	Output or Static OOB data

Table 2.464. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x09	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 *da
ta_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;
};
```

2.17.1.42 cmd_mesh_prov_oob_pkey_rsp

This command is used to respond to prov_oob_pkey_request

Table 2.465. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x01	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x08	method	Message ID
4	uint8array	pkey	Public Key read out-of-band

Table 2.466. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x08	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 *pk
ey_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;
};
```

2.17.1.43 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. 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 2.467. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.468. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x02	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_prov_provision_device_rsp_t *gecko_cmd_mesh_prov_provision_device(uint8 network_id, uint
8 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 2.469. Events Generated

Event	Description
mesh_prov_device_provisioned	Device provisioned successfully.
mesh_prov_provisioning_failed	Provisioning a device failed.

2.17.1.44 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. 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 2.470. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.471. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x15	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.472. Events Generated

Event	Description
mesh_prov_device_provisioned	Device provisioned successfully.
mesh_prov_provisioning_failed	Provisioning a device failed.

2.17.1.45 cmd_mesh_prov_relay_get

Get node relay retransmission state. The node will respond with a subscription status event.

Table 2.473. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.474. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x27	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.475. Events Generated

Event	Description
mesh_prov_relay_status	Node relay state report.

2.17.1.46 cmd_mesh_prov_relay_set

Set node relay retransmission state. The node will respond with a relay status event.

Table 2.476. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x07	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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: • **0x00:** Relaying disabled • **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 2.477. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x28	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.478. Events Generated

Event	Description
mesh_prov_relay_status	Node relay state report.

2.17.1.47 cmd_mesh_prov_reset_node

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 a key refresh, with the removed node blacklisted, should be done.

Table 2.479. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 2.480. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x29	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.481. Events Generated

Event	Description
mesh_prov_node_reset	A node has reset itself.

2.17.1.48 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 2.482. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x00	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x01	method	Message ID

Table 2.483. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x01	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.484. Events Generated

Event	Description
mesh_prov_unprov_beacon	Unprovisioned beacon seen.

2.17.1.49 cmd_mesh_prov_set_config

Set a configration state value of a Node. Node Configuration Server model state contains a number of node-wide values (for instance, Node's default TTL value) which are represented as single bytes; they can be modified with this command. See the list of configuration states for reference. Setting the more complex states should be done using the commands dedicated for the purpose as this command accepts only raw binary data as the value to set. Node response is reported with an configuration status event.

Table 2.485. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x07	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x06	method	Message ID
4-5	uint16	address	Unicast address of the target Node's primary element
6-7	uint16	id	The State to manipulate
8-9	uint16	netkey_index	Ignored for node-wide States.
10	uint8array	value	Raw binary value

Table 2.486. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x06	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 n
etkey_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 2.487. Events Generated

Event	Description
mesh_prov_config_status	Remote Status response to a Get/Set request.

2.17.1.50 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 thus be shut out of the network.

Table 2.488. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0d	method	Message ID
4-5	uint16	key	Network key index
6	uint8	status	Nonzero for blacklisted node
7	uint8array	uuid	UUID of the Device

Table 2.489. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0d	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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(uint1
6 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;
};
```

2.17.1.51 cmd_mesh_prov_set_oob_requirements

Set the OOB requirements for devices to be Provisioned

Table 2.490. Command

Byte	Type	Name	Description
0	0x20	hilen	Message type: Command
1	0x08	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0a	method	Message ID
4	uint8	public_key	Zero to not use OOB Public Key
5	uint8	auth_methods	Allowed OOB authentication methods The value is a bitmap so that multiple methods can be allowed. Valid values are as follows: • **Bit 0:** No OOB is allowed • **Bit 1:** Static OOB is allowed • **Bit 2:** Input OOB is allowed • **Bit 3:** Output OOB is allowed
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
11	uint8	max_size	Maximum Input/Output OOB size

Table 2.491. Response

Byte	Type	Name	Description
0	0x20	hilen	Message type: Response
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x0a	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_ke
y, 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;
};
```

2.17.2 mesh_prov events

2.17.2.1 evt_mesh_prov_appkey_list

Application key list event.

Table 2.492. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.2 evt_mesh_prov_appkey_list_end

Application key list terminator event.

Table 2.493. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x10	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.2.3 evt_mesh_prov_config_status

Remote Status response to a Get/Set request.

Table 2.494. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x06	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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, the data field should be ignored.
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;
};
```

2.17.2.4 evt_mesh_prov_dcd_status

This event carries the device composition data of a node

Table 2.495. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x11	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x04	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.2.5 evt_mesh_prov_ddb_list

Provisioner's device database list entry

Table 2.496. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x00	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.6 evt_mesh_prov_device_provisioned

Device provisioned successfully.

Table 2.497. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.7 evt_mesh_prov_friend_timeout_status

Node relay state report.

Table 2.498. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x08	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.8 evt_mesh_prov_heartbeat_publication_status

Node heartbeat status, generated in response to a get heartbeat publication state or set heartbeat publication state request.

Table 2.499. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x0d	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.9 evt_mesh_prov_heartbeat_subscription_status

Node heartbeat subscription report.

Table 2.500. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x0c	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.10 evt_mesh_prov_initialized

Provisioner initialized and operational.

Table 2.501. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x07	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x00	method	Message ID
4	uint8	networks	Number of network keys 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;
};
```

2.17.2.11 evt_mesh_prov_key_refresh_complete

Key refresh for a network key has completed

Table 2.502. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x16	method	Message ID
4-5	uint16	key	Network key index
6-7	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.2.12 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 2.503. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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 node.

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;
};
```

2.17.2.13 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 2.504. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.14 evt_mesh_prov_model_pub_status

Table 2.505. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x10	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x13	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	elem_address	Unicast address of the element containing the model being queried.
8-9	uint16	vendor_id	Vendor ID of model being queried. Returns 0xffff for Bluetooth SIG models.
10-11	uint16	model_id	Model ID of the model being queried.
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: • **Bits 0..5:** Step count • **Bits 6..7:** Step resolution: • 00: 100 milliseconds • 01: 1 second • 10: 10 seconds • 11: 10 minutes
18	uint8	retrans	Unsigned 8-bit integer
19	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.

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;
};
```

2.17.2.15 evt_mesh_prov_model_sub_addr

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 2.506. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x08	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x17	method	Message ID
4-5	uint16	elem_address	Unicast address of the element containing the model being queried.
6-7	uint16	vendor_id	Vendor ID of model being queried. Returns 0xffff for Bluetooth SIG models.
8-9	uint16	model_id	Model ID of the model being queried.
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;
};
```

2.17.2.16 evt_mesh_prov_model_sub_addr_end

This event terminates model subscription list result reporting.

Table 2.507. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x08	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x18	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	elem_address	Unicast address of the element containing the model being quer-ied.
8-9	uint16	vendor_id	Vendor ID of model being queried. Returns 0xffff for Bluetooth SIG models.
10-11	uint16	model_id	Model ID of the model being queried.

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;
};
```

2.17.2.17 evt_mesh_prov_network_list

Network key list event.

Table 2.508. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.18 evt_mesh_prov_network_list_end

Network key list terminator event.

Table 2.509. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x12	method	Message ID
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.17.2.19 evt_mesh_prov_node_reset

A node has reset itself.

Table 2.510. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.20 evt_mesh_prov_oob_auth_request

The Provisioner needs the Device's Output or Static Data. It should be provided using prov_oob_auth_rsp.

Table 2.511. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x04	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.21 evt_mesh_prov_oob_display_input

Random OOB input data was generated; this should be displayed to and input with the Deivce.

Table 2.512. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x03	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x08	method	Message ID
4	uint8	input_action	Input Action type
5	uint8	input_size	Number of digits
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;
};
```

2.17.2.22 evt_mesh_prov_oob_pkey_request

The Provisioner needs the OOB public key of the Device with given UUID. The key should be input using prov_oob_pkey_rsp.

Table 2.513. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x01	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.2.23 evt_mesh_prov_provisioning_failed

Provisioning a device failed.

Table 2.514. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x02	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x01	method	Message ID
4	uint8	reason	Reason for failure
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;
};
```

2.17.2.24 evt_mesh_prov_relay_status

Node relay state report.

Table 2.515. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x07	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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: • **0x00:** Relaying disabled • **0x01:** Relaying enabled • **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;
};
```

2.17.2.25 evt_mesh_prov_unprov_beacon

Unprovisioned beacon seen.

Table 2.516. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x0f	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh Provisioner
3	0x03	method	Message ID
4-5	uint16	oob_capabilities	OOB capabilities bitfield. Indicates the means by which out-of-band provisioning data may be retrieved. See for details.
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 shall be ignored.
10	uint8	bearer	Bearer on which the beacon was seen. Values are as follows: • **0:** PB-ADV • **1:** PB-GATT
11-16	bd_addr	address	Address of the device beaconin
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
{
  uint16 oob_capabilities;,
  uint32 uri_hash;,
  uint8 bearer;,
  bd_addr address;,
  uint8 address_type;,
  uint8array uuid;
};
```

2.17.2.26 evt_mesh_prov_uri

URI advertisement received from a nearby device.

Table 2.517. Event

Byte	Type	Name	Description
0	0xa0	hilen	Message type: Event
1	0x05	lolen	Minimum payload length
2	0x15	class	Message class: Bluetooth Mesh 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;
};
```

2.17.3 mesh_prov defines

2.17.3.1 define_mesh_prov_oob_capabilities

OOB capability bitmask constants

Table 2.518. 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

2.18 Bluetooth Mesh Proxy Connections (mesh_proxy)

Bluetooth Mesh functions for GATT proxy connections

2.18.1 mesh_proxy commands

2.18.1.1 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 2.519. 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	Connection 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 2.520. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.18.1.2 cmd_mesh_proxy_connect

Start connecting a proxy client to a proxy server. Once the connection is complete, a connection established event will be generated. LE-connection must be opened prior to opening proxy connection

Table 2.521. 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 2.522. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-9	uint32	handle	If a connection attempt is successfully initiated a valid connection 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;
};
```

2.18.1.3 cmd_mesh_proxy_deny

Block messages destined to 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 2.523. 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	Connection 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 2.524. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.18.1.4 cmd_mesh_proxy_disconnect

Disconnect. This call can be used also for a connection which is not yet fully formed.

Table 2.525. 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	Connection handle

Table 2.526. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.18.1.5 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 2.527. 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	Connection 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 2.528. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 ty
pe, 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;
};
```

2.18.2 mesh_proxy events

2.18.2.1 evt_mesh_proxy_connected

Indication that a connection has been successfully formed.

Table 2.529. 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	Connection handle

C Functions

```
/* Event id */
gecko_evt_mesh_proxy_connected_id

/* Event structure */
struct gecko_msg_mesh_proxy_connected_evt_t
{
  uint32 handle;
};
```

2.18.2.2 evt_mesh_proxy_disconnected

Indication that a connection has been disconnected or a connection attempt failed.

Table 2.530. 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	Connection 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;
};
```

2.18.2.3 evt_mesh_proxy_filter_status

Proxy status report event

Table 2.531. 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	Connection 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;
};
```

2.19 Bluetooth Mesh GATT Proxy Client (mesh_proxy_client)

This class is used to initialize the GATT Proxy client-side functionality. Mesh proxy commands are in the mesh_proxy class. This class exists solely for the purpose of allowing 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.

2.20 Bluetooth Mesh GATT Proxy Server (mesh_proxy_server)

This class is used to initialize the GATT Proxy server-side functionality. This class exists solely for the purpose of allowing 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.

2.21 Bluetooth Mesh test utilities (mesh_test)

These commands are meant for development and testing. They are not be used in production software.

2.21.1 mesh_test commands

2.21.1.1 cmd_mesh_test_add_local_key

Add a network or application key locally.

Table 2.532. 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	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 2.533. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 k
ey, 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;
};
```

2.21.1.2 cmd_mesh_test_add_local_model_sub

Add an address to a local model's subscription list.

Table 2.534. 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 2.535. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.3 cmd_mesh_test_add_local_model_sub_va

Add a virtual address to a local model's subscription list.

Table 2.536. 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 2.537. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.4 cmd_mesh_test_bind_local_model_app

Bind a Model to an Appkey locally.

Table 2.538. 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 2.539. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_ind
ex, 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;
};
```

2.21.1.5 cmd_mesh_test_del_local_key

Delete a network or application key locally.

Table 2.540. 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 2.541. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_in
dex);

/* Response id */
gecko_rsp_mesh_test_del_local_key_id

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

2.21.1.6 cmd_mesh_test_del_local_model_sub

Remove an address from a local Model's subscription list.

Table 2.542. 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 2.543. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.7 cmd_mesh_test_del_local_model_sub_va

Remove a virtual address from a local model's subscription list.

Table 2.544. 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 2.545. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.8 cmd_mesh_test_get_element_seqnum

Get current sequence number of an element

Table 2.546. 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 2.547. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-9	uint32	seqnum	Current sequence number of the element Value is to be ignored if the result code indicates an error (for instance in the case 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;
};
```

2.21.1.9 cmd_mesh_test_get_ivupdate_test_mode

Table 2.548. 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 2.549. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8	mode	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;
};
```

2.21.1.10 cmd_mesh_test_get_local_config

Get the value of a state in the configuration server model; this should be used for testing and debugging purposes only.

Table 2.550. 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 2.551. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 netke
y_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;
};
```

2.21.1.11 cmd_mesh_test_get_local_heartbeat_publication

Get heartbeat publication state of a local node.

Table 2.552. 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 2.553. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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.
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_publi
cation();

/* 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;
};
```

2.21.1.12 cmd_mesh_test_get_local_heartbeat_subscription

Get local node heartbeat subscription state

Table 2.554. 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 2.555. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_subs
cription();

/* 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;
};
```

2.21.1.13 cmd_mesh_test_get_local_model_pub

Get a local model's publication address, key, and parameters.

Table 2.556. 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 2.557. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	appkey_index	The application key index used for the application messages pub-lished
8-9	uint16	pub_address	The address published to
10	uint8	ttl	Time-to-Live value for published messages
11	uint8	period	Unsigned 8-bit integer
12	uint8	retrans	Unsigned 8-bit integer
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;
};
```

2.21.1.14 cmd_mesh_test_get_local_model_sub

Get all entries in a local model's subscription list.

Table 2.558. 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 2.559. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8array	addresses	List of 16-bit Mesh addresses; empty if not subscribed to any address. Ignore if result code is nonzero.

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;
};
```

2.21.1.15 cmd_mesh_test_get_nettx

Get the network transmit state of a node.

Table 2.560. 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 2.561. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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) millisec-onds; e.g. 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;
};
```

2.21.1.16 cmd_mesh_test_get_relay

Table 2.562. 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 2.563. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 reransmit 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;
};
```

2.21.1.17 cmd_mesh_test_send_beacons

Table 2.564. 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 2.565. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.18 cmd_mesh_test_set_adv_scan_params

Set non-default advertisement and scanning parameters used in mesh communications. Note that this command needs to be called before node initialization or Provisioner initialization for the settings to take effect.

Table 2.566. 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 0.625ms. Default value is 1 (0.625ms).
6-7	uint16	adv_interval_max	Maximum advertisement interval. Value is in units of 0.625ms. 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 0.625ms. Range: 0x0004 to 0x4000 (time range of 2.5ms to 10.24s). Default value is 160 (100ms).
13-14	uint16	scan_window	Scan window. Value is in units of 0.625ms. Must be equal to or less than the scan interval

Table 2.567. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_interv
al_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;
};
```

2.21.1.19 cmd_mesh_test_set_ivupdate_state

Table 2.568. 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 2.569. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.20 cmd_mesh_test_set_ivupdate_test_mode

Table 2.570. 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 2.571. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.21 cmd_mesh_test_set_local_config

Set a state to a value in the local configuration server model; this should be used for testing and debugging purposes only.

Table 2.572. 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 2.573. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 netke
y_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;
};
```

2.21.1.22 cmd_mesh_test_set_local_heartbeat_publication

Set heartbeat publication state of a local node.

Table 2.574. 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 cannot 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: • **0x00:** Heartbeat messages are not sent • **0x01 .. 0x11:** Node shall send 2^(n-1) heartbeat messages • **0x12 .. 0xfe:** Prohibited • **0xff:** Hearbeat messages are sent indefinitely
7	uint8	period_log	Heartbeat publication period setting. Valid values are as follows: • **0x00:** Heartbeat messages are not sent • **0x01 .. 0x11:** Node shall send a heartbeat message every 2^(n-1) seconds • **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: • **Bit 0:** Relay feature • **Bit 1:** Proxy feature • **Bit 2:** Friend feature • **Bit 3:** Low power feature Remaining bits are reserved for future use.
11-12	uint16	publication_netkey_index	Index of the network key used to encrypt heartbeat messages.

Table 2.575. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_test_set_local_heartbeat_publication_rsp_t *gecko_cmd_mesh_test_set_local_heartbeat_publi
cation(uint16 publication_address, uint8 count_log, uint8 period_log, uint8 ttl, uint16 features, uint16 public
ation_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;
};
```

2.21.1.23 cmd_mesh_test_set_local_heartbeat_subscription

Set local node heartbeat subscription parameters. Normally heartbeat subscription is controlled by the Provisioner.

Table 2.576. 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: • **0x00:** Heartbeat messages are not received • **0x01 .. 0x11:** Node shall receive heartbeat messages for 2^(n-1) seconds • **0x12 .. 0xff:** Prohibited

Table 2.577. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_test_set_local_heartbeat_subscription_rsp_t *gecko_cmd_mesh_test_set_local_heartbeat_subs
cription(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;
};
```

2.21.1.24 cmd_mesh_test_set_local_model_pub

Set a local model's publication address, key, and parameters.

Table 2.578. 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	Unsigned 8-bit integer
16	uint8	retrans	Unsigned 8-bit integer
17	uint8	credentials	Friendship credentials flag

Table 2.579. 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
4-5	uint16	result	Result code • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 re
trans, 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;
};
```

2.21.1.25 cmd_mesh_test_set_local_model_pub_va

Set a model's publication virtual address, key, and parameters.

Table 2.580. 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 model being configured. Use 0xffff for Bluetooth SIG models.
10-11	uint16	model_id	Model ID of the model being configured.
12	uint8	ttl	Publication time-to-live value
13	uint8	period	Unsigned 8-bit integer
14	uint8	retrans	Unsigned 8-bit integer
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 2.581. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 c
redentials, 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;
};
```

2.21.1.26 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 2.582. 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; e.g. for a value of 2 the interval would be between 30 and 40 milliseconds. Range: 0-31.

Table 2.583. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.27 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 setting done by the Provisioner.

Table 2.584. 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	Setting indicating whether the relay functionality is enabled on the node (1) or not (0); value indicating disabled (2) cannot be set.
5	uint8	count	Number of relay transmissions beyond the initial one. Range: 0-7.
6	uint8	interval	Relay reransmit interval steps. The interval between transmissions is 10*(1+steps) milliseconds. Range: 0-31.

Table 2.585. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 int
erval);

/* Response id */
gecko_rsp_mesh_test_set_relay_id

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

2.21.1.28 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 2.586. 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_tim-er_ms	Maximum timeout before a transaction expires, regardless of oth-er parameters. Value is in milliseconds. Default = 10000 (10 sec-onds).
8-11	uint32	pend-ing_ack_base_ms	Base time to wait at the receiver before sending a transport layer acknowledgement. Value is in milliseconds. Default = 150.
12-15	uint32	pend-ing_ack_mul_ms	TTL multiplier to add to the base acknowledgement timer. Value is in milliseconds. Default = 50.
16-19	uint32	wait_for_ack_base_ms	Base time to wait for an acknowledgement 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 2.587. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_m
s, 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;
};
```

2.21.1.29 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 2.588. 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 2.589. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.30 cmd_mesh_test_unbind_local_model_app

Remove a binding between a Model and an Appkey locally.

Table 2.590. 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 2.591. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.1.31 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 normal 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 2.592. 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	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 2.593. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.21.2 mesh_test events

2.21.2.1 evt_mesh_test_local_heartbeat_subscription_complete

Event indicating heartbeat subscription period is over

Table 2.594. 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;
};
```

2.22 Bluetooth Mesh Vendor Model (mesh_vendor_model)

Vendor model API provides functionality to send and receive vendor specific messages. Throughout the API the model being manipulated is identified by its element address, vendor ID and model ID. The API has functions for sending, receiving, and publishing messages; it is up to the application to implement any more 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).

2.22.1 mesh_vendor_model commands

2.22.1.1 cmd_mesh_vendor_model_clear_publication

Clear vendor model publication message. Clearing the model publication message disables model publishing; it can be re-enabled by defining the publication message using the set vendor model publication command.

Table 2.595. 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 2.596. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_vendor_model_clear_publication_rsp_t *gecko_cmd_mesh_vendor_model_clear_publication(uint1
6 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;
};
```

2.22.1.2 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 2.597. 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 2.598. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 v
endor_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;
};
```

2.22.1.3 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. It is necessary to define the opcodes the model is able to receive at initialization. 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 2.599. 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 (nonzero) or not (zero).
11	uint8array	opcodes	Array of opcodes the model can handle.

Table 2.600. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 vendo
r_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;
};
```

2.22.1.4 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 2.601. 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 2.602. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_vendor_model_publish_rsp_t *gecko_cmd_mesh_vendor_model_publish(uint16 elem_index, uint1
6 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;
};
```

2.22.1.5 cmd_mesh_vendor_model_send

Send vendor specific data Note that due to 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 nonzero 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 2.603. 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. 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 nonzero if the message should affect only devices in the immediate radio neighborhood.
16	uint8	opcode	Message opcode
17	uint8	final	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 2.604. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 vendo
r_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;
};
```

2.22.1.6 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 periodc publishing) or explicitly (see vendor model publish command. Note that due to 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 nonzero 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 2.605. 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	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 2.606. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_mesh_vendor_model_set_publication_rsp_t *gecko_cmd_mesh_vendor_model_set_publication(uint16 el
em_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;
};
```

2.22.2 mesh_vendor_model events

2.22.2.1 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 due to 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 nonzero value. It is up to the application to concatenate the messages into a single buffer if it is necessary.

Table 2.607. 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; 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 nonzero, indicates that the received message was not relayed (TTL was zero); this means the devices are within direct radio range of each other.
18	uint8	opcode	Message opcode
19	uint8	final	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;
};
```

2.23 Security Manager (sm)

The commands in this section are used to manage Bluetooth security, including commands for starting and stopping encryption and commands for management of all bonding operations.

The following procedure can be 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_open 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 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 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 user to enter a passkey during the process. See events sm_passkey_display and sm_passkey_request for more information.

2.23.1 sm commands

2.23.1.1 cmd_sm_bonding_confirm

This command can be used for accepting or rejecting bonding request.

Table 2.608. 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	Accept bonding request. Values: • **0:** Reject • **1:** Accept bonding request

Table 2.609. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.2 cmd_sm_configure

This command can be used to configure security requirements and I/O capabilities of the system.

Table 2.610. 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.Bit 0: • **0:** Allow bonding without MITM protection • **1:** Bonding requires MITM protection Bit 1: • **0:** Allow encryption without bonding • **1:** Encryption requires bonding. Note that this setting will also enable bonding. Bit 2: • **0:** Allow bonding with legacy pairing • **1:** Secure connections only Bit 3: • **0:** Bonding request does not need to be confirmed • **1:** Bonding requests need to be confirmed. Received bonding requests are notified with sm_confirm_bonding events. Bit 4 to 7: ReservedDefault value: 0x00
5	uint8	io_capabilities	I/O Capabilities. See link

Table 2.611. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.3 cmd_sm_delete_bonding

This command can be used to delete specified bonding information from Persistent Store.

Table 2.612. 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 2.613. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.4 cmd_sm_delete_bondings

This command can be used to delete all bonding information from Persistent Store.

Table 2.614. 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 2.615. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.5 cmd_sm_enter_passkey

This command can be used to enter a passkey after receiving a passkey request event.

Table 2.616. 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 2.617. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.6 cmd_sm_increase_security

This command can be used to enhance the security of a connection to current security requirements. On an unencrypted connection, this will encrypt the connection and will also perform bonding if requested by both devices. On an encrypted connection, this will cause the connection re-encrypted.

Table 2.618. 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 2.619. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.620. Events Generated

Event	Description
le_connection_parameters	This event is triggered after increasing security has been completed successfully, and indicates the latest security mode of the connection.
sm_bonded	This event is triggered if pairing or bonding was performed in this operation and the result is success.
sm_bonding_failed	This event is triggered if pairing or bonding was performed in this operation and the result is failure.

2.23.1.7 cmd_sm_list_all_bondings

This command can be used to list all bondings stored in the bonding database. Bondings are reported by using the sm_list_bonding_entry event for each bonding and the report is ended with sm_list_all_bondings_complete event. Recommended to be used only for debugging purposes, because reading from the Persistent Store is relatively slow.

Table 2.621. 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 2.622. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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 2.623. Events Generated

Event	Description
sm_list_bonding_entry	This event is triggered by the command sm_list_all_bondings if bondings exist in the local database.
sm_list_all_bondings_complete	This event is triggered by the sm_list_all_bondings and follows sm_list_bonding_entry events.

2.23.1.8 cmd_sm_passkey_confirm

This command can be used for accepting or rejecting reported confirm value.

Table 2.624. 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	Accept confirm value. Values: • **0:** Reject • **1:** Accept confirm value

Table 2.625. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.9 cmd_sm_set_bondable_mode

This command can be used to set whether the device should accept new bondings. By default, the device does not accept new bondings.

Table 2.626. 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: • **0:** New bondings not accepted • **1:** Bondings allowed Default value: 0

Table 2.627. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.10 cmd_sm_set_debug_mode

This command can be used to set Security Manager in debug mode. In this mode the secure connections bonding uses debug keys, so that the encrypted packet can be opened by Bluetooth protocol analyzer. To disable the debug mode, you need to restart the device.

Table 2.628. 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 2.629. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.11 cmd_sm_set_oob_data

This command can be used to set the OOB data (out-of-band encryption data) for legacy pairing for a device. The OOB data may be, for example, a PIN code exchanged over an alternate path like NFC. The device will not allow any other kind of bonding if OOB data is set. The OOB data cannot be set simultaneously with secure connections OOB data.

Table 2.630. 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 2.631. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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_dat
a);

/* Response id */
gecko_rsp_sm_set_oob_data_id

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

2.23.1.12 cmd_sm_set_passkey

This command can be used to enter a fixed passkey which will be used in the sm_passkey_display event.

Table 2.632. 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 2.633. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.23.1.13 cmd_sm_set_sc_remote_oob_data

This command can be used to 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 2.634. 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 the OOB data and last 16-bytes the confirm value. To clear OOB data, send a zero-length array.

Table 2.635. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, cons
t 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;
};
```

2.23.1.14 cmd_sm_store_bonding_configuration

This command can be used to set maximum allowed bonding count and bonding policy. The default value is maximum number of bondings supported.

Table 2.636. 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 14
5	uint8	policy_flags	Bonding policy. Values: • **0:** If database is full, new bonding attempts will fail • **1:** New bonding will overwrite the oldest existing bonding • **2:** New bonding will overwrite longest time ago used existing bonding

Table 2.637. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

BGLIB C API

```
/* Function */
struct gecko_msg_sm_store_bonding_configuration_rsp_t *gecko_cmd_sm_store_bonding_configuration(uint8 max_bondi
ng_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;
};
```

2.23.1.15 cmd_sm_use_sc_oob

This command can be used to enable the use of OOB data (out-of-band encryption data) for a device for secure connections pairing. The enabling will genarate 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 kind of bonding if OOB data is set. The secure connections OOB data cannot be enabled simultaneously with legacy pairing OOB data.

Table 2.638. 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: • **0:** disable • **1:** enable

Table 2.639. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6	uint8array	oob_data	OOB data. 32-byte array. First 16-bytes contain randomly generated OOB data and 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;
};
```

2.23.2 sm events

2.23.2.1 evt_sm_bonded

This event is triggered after the pairing or bonding procedure has been successfully completed.

Table 2.640. 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: • **0xff:** Pairing completed without bonding - the pairing key will be discarded after disconnection. • **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;
};
```

2.23.2.2 evt_sm_bonding_failed

This event is triggered if the pairing or bonding procedure has failed.

Table 2.641. 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;
};
```

2.23.2.3 evt_sm_confirm_bonding

This event indicates a request to display that new bonding request has been received to the user and for the user to confirm the request. Use the command sm_bonding_confirm to accept or reject the bonding request.

Table 2.642. 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. • NOTE! When the bonding handle is anything else than -1 there is already existing bonding for this connection. Overwriting existing bonding is 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;
};
```

2.23.2.4 evt_sm_confirm_passkey

This event indicates a request to display the passkey to the user and for the user to confirm the displayed passkey. Use the command sm_passkey_confirm to accept or reject the displayed passkey.

Table 2.643. 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. • NOTE! When displaying the passkey to the user, prefix the number with zeros in order to obtain a 6 digit number • Example: Passkey value is 42 • Number to display to 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;
};
```

2.23.2.5 evt_sm_list_all_bondings_complete

This event is triggered by the sm_list_all_bondings and follows sm_list_bonding_entry events.

Table 2.644. 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
{
};
```

2.23.2.6 evt_sm_list_bonding_entry

This event is triggered by the command sm_list_all_bondings if bondings exist in the local database.

Table 2.645. 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 index of bonding data
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;
};
```

2.23.2.7 evt_sm_passkey_display

This event indicates a request to display the passkey to the user.

Table 2.646. 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. • NOTE! When displaying the passkey to the user, prefix the number with zeros in order to obtain a 6 digit number • Example: Passkey value is 42 • Number to display to 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;
};
```

2.23.2.8 evt_sm_passkey_request

This event indicates a request for the user to enter the passkey displayed on the remote device. Use the command sm_enter_passkey to input the passkey value.

Table 2.647. 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;
};
```

2.23.3 sm enumerations

2.23.3.1 enum_sm_bonding_key

These values define the bonding information of the bonded device stored in persistent store.

Table 2.648. 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

2.23.3.2 enum_sm_io_capability

These values define the security management related I/O capabilities supported by the device

Table 2.649. 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

2.24 System (system)

The commands and events in this class can be used to access and query the local device.

2.24.1 system commands

2.24.1.1 cmd_system_get_bt_address

This command can be used to read the Bluetooth public address used by the device.

Table 2.650. 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 2.651. 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;
};
```

2.24.1.2 cmd_system_get_counters

Get packet and error counters

Table 2.652. 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 parameter value is nonzero

Table 2.653. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes
6-7	uint16	tx_packets	Number of successfully transmitted packets
8-9	uint16	rx_packets	Number of successfully received packets
10-11	uint16	crc_errors	Number of received packets with CRC errors
12-13	uint16	failures	Number of radio failures like aborted tx/rx packets, scheduling failures, etc

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;
};
```

2.24.1.3 cmd_system_get_random_data

This command can be used to get random data up to 16 bytes.

Table 2.654. 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 2.655. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.24.1.4 cmd_system_halt

This command forces radio to idle state and allows device to sleep. Advertising, scanning, connections and software timers are halted by this commands. Halted operations are resumed by calling this command with parameter 0. Connections stay alive if system is re-sumed before connection supervision timeout.

NOTE: Software timer is also halted. Hardware interrupts are the only way to wake up from energy mode 2 when system is halted.

Table 2.656. 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: • **1:** halt • **0:** resume

Table 2.657. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.24.1.5 cmd_system_hello

This command does not trigger any event but the response to the command is used to verify that communication between the host and the device is working.

Table 2.658. 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 2.659. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.24.1.6 cmd_system_reset

This command can be used to reset the system. It 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 2.660. 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: • **0:** Normal reset • **1:** Boot to UART DFU mode • **2:** Boot to OTA DFU mode

BGLIB C API

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

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

Table 2.661. Events Generated

Event	Description
system_boot	Sent after the device has booted into normal mode
dfu_boot	Sent after the device has booted into UART DFU mode

2.24.1.7 cmd_system_set_bt_address

This command can be used to 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 it will be effective in the next system reboot. The stack treats 00:00:00:00:00:00 and ff:ff:ff:ff:ff:ff as invalid addresses. Thus passing one of them into this command will cause the stack to use the default address in the next system reboot.

Table 2.662. 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 2.663. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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;
};
```

2.24.1.8 cmd_system_set_device_name

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

Table 2.664. 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: • **0:** OTA device name
5	uint8array	name	Device name

Table 2.665. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the 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, con
st 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;
};
```

2.24.1.9 cmd_system_set_tx_power

This command can be used to set the maximum TX power for Bluetooth. The returned value in the response is the selected maximum output power level after applying RF path compensation. If the GATT server contains a Tx Power service, the Tx Power Level attribute of the service will be updated accordingly.

The stack will choose the maximum allowed for Bluetooth connections. The chosen power level may be less than the specified value if the device does not meet the power requirements. For Bluetooth connections the maximum TX power will be limited to 10 dBm if Adaptive Frequency Hopping (AFH) is not enabled.

NOTE: This command should not be used while advertising, scanning or during connection.

Table 2.666. 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 0.1dBm steps, for example the value of 10 is 1dBm and 55 is 5.5dBm

Table 2.667. 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;
};
```

2.24.2 system events

2.24.2.1 evt_system_awake

This event indicates that the device has woken up from sleep mode.

NOTE: Stack does not generate this event by itself as sleep and wakeup are managed in applications. If this event is needed, the application should call function gecko_send_system_awake() to signal stack to send this event.

Table 2.668. 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
{
};
```

2.24.2.2 evt_system_boot

This event indicates the device has started and the radio is ready. This even carries the firmware build number and other SW and HW identification codes.

Table 2.669. 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;
};
```

2.24.2.3 evt_system_error

This event indicates errors.

Table 2.670. 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	Standard code of the error just happened.
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;
};
```

2.24.2.4 evt_system_external_signal

This event indicates external signals have been received. External signals are generated from native application.

Table 2.671. 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;
};
```

2.24.2.5 evt_system_hardware_error

This event indicates that hardware related errors occurred.

Table 2.672. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the Error codes

C Functions

```
/* Event id */
gecko_evt_system_hardware_error_id

/* Event structure */
struct gecko_msg_system_hardware_error_evt_t
{
  uint16 status;
};
```

2.25 testing commands (test)

2.25.1 test commands

2.25.1.1 cmd_test_dtm_end

This command can be used to 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 2.673. 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 2.674. 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 2.675. Events Generated

Event	Description
test_dtm_completed	This event is received when the command is processed.

2.25.1.2 cmd_test_dtm_rx

This command can be used to start a receiver test. The test is meant to be used against a separate Bluetooth tester device. When the command is processed by the radio, a test_dtm_completed event is triggered. This event indicates if the test started successfully.

Parameter **phy** specifies which PHY is used to receive the packets. All devices support at least the 1M PHY.

The test may be stopped using the test_dtm_end command. This will trigger another test_dtm_completed event, which carries the number of packets received during the test.

Table 2.676. 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 **Range:** 0-39 Channel is (F - 2402) / 2, where F is frequency in MHz
5	uint8	phy	PHY to use

Table 2.677. 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 2.678. Events Generated

Event	Description
test_dtm_completed	This event is received when the command is processed.

2.25.1.3 cmd_test_dtm_tx

This command can be used to start a transmitter test. The test is meant to be used against a separate Bluetooth tester device. When the command is processed by the radio, a test_dtm_completed event is triggered. This event indicates if 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. Parameter **phy** specifies which PHY is used to transmit the packets. All devices support at least the 1M PHY. There is also a special packet type, **test_pkt_carrier**, which can be used to transmit continuous unmodulated carrier. The **length** field is ignored in this mode.

The test may be stopped using the test_dtm_end command.

Table 2.679. 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 **Range:** 0-255
6	uint8	channel	Bluetooth channel **Range:** 0-39 Channel is (F - 2402) / 2, where F is frequency in MHz
7	uint8	phy	PHY to use

Table 2.680. 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, uint
8 phy);

/* Response id */
gecko_rsp_test_dtm_tx_id

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

Table 2.681. Events Generated

Event	Description
test_dtm_completed	This event is received when the command is processed.

2.25.2 test events

2.25.2.1 evt_test_dtm_completed

This event 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 2.682. 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 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;
};
```

2.25.3 test enumerations

2.25.3.1 enum_test_packet_type

Test packet types

Number 3 corresponding to unmodulated carrier packet type is deprecated. Valid number is 254.

Table 2.683. 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
3	test_pkt_carrier_deprecated	Unmodulated carrier - deprecated
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

2.25.3.2 enum_test_phy

Test PHY types

Table 2.684. 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

2.26 User messaging (user)

This class provides one command and one event which can be utilized by a NCP host and target to implement a communication mechanism with own proprietary protocol. An application has the full responsibility on deciding whether and how the command and event are used. The stack does not produce or consume any messages belonging to this class.

2.26.1 user commands

2.26.1.1 cmd_user_message_to_target

This command can be used by an NCP host to send a message to the target application on device.

Table 2.685. 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 2.686. 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 • **0:** success • **Non-zero:** an error occurred For other values refer to the
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 *da
ta_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;
};
```

2.26.2 user events

2.26.2.1 evt_user_message_to_host

This event can be used by the target application on device to send a message to NCP host.

Table 2.687. 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;
};
```

2.27 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
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
0x0184	invalid_command	Command was not recognized
0x0185	timeout	Command or Procedure failed due to timeout

Code	Name	Description
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

- **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
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

Code	Name	Description
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
0x0202	unknown_connection_identifier	Connection does not exist, or connection open request was cancelled.
0x0204	page_timeout	The Page Timeout error code indicates that a page timed out because of the Page Timeout configuration parameter.
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.
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.

Code	Name	Description
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
0x0219	unknown_lmp_pdu	The Controller has received an unknown LMP OpCode.
0x021a	unsupported_remote_feature	The remote device does not support the feature associated with the issued command or LMP PDU.
0x021b	sco_offset_rejected	The offset requested in the LMP_SCO_link_req PDU has been rejected.
0x021c	sco_interval_rejected	The interval requested in the LMP_SCO_link_req PDU has been rejected.
0x021d	sco_air_mode_rejected	The air mode requested in the LMP_SCO_link_req PDU has been rejected.
0x021e	invalid_lmp_parameters	Some LMP PDU / LL Control PDU parameters were invalid.
0x021f	unspecified_error	No other error code specified is appropriate to use.
0x0220	unsupported_lmp_parameter_value	An LMP PDU or an LL Control PDU contains at least one parameter value that is not supported by the Controller at this time.
0x0221	role_change_not_allowed	Controller will not allow a role change at this time.
0x0222	ll_response_timeout	Connection terminated due to link-layer procedure timeout.
0x0223	lmp_error_transaction_collision	LMP transaction has collided with the same transaction that is already in progress.
0x0224	lmp_pdu_not_allowed	Controller sent an LMP PDU with an OpCode that was not allowed.
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.

Code	Name	Description
0x0227	requested_qos_not_supported	The requested Quality of Service is not supported.
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.
0x022c	qos_unacceptable_parameter	The specified quality of service parameters could not be accepted at this time, but other parameters may be acceptable.
0x022d	qos_rejected	The specified quality of service parameters cannot be accepted and QoS negotiation should be terminated.
0x022e	channel_assesment_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.
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.
0x0232	role_switch_pending	Role Switch is pending. This can be used when an HCI command or LMP PDU cannot be accepted because of a pending role switch. This can also be used to notify a peer device about a pending role switch.
0x0234	reserved_slot_violation	The current Synchronous negotiation was terminated with the negotiation state set to Reserved Slot Violation.
0x0235	role_switch_failed	role switch was attempted but it failed and the original piconet structure is restored. The switch may have failed because the TDD switch or piconet switch failed.
0x0236	extended_inquiry_response_too_large	The extended inquiry response, with the requested requirements for FEC, is too large to fit in any of the packet types supported by the Controller.
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	directed_advertising_timeout	Directed 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.

Code	Name	Description
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.

- **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.
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 reestablish 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.

- **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

Code	Name	Description
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 encountered 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
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	When this is returned in a BGAPI response, the application tried to read or write the value of a user attribute from the GATT database.

- **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.

- **Filesystem errors**

Code	Name	Description
0x0901	file_not_found	File not found

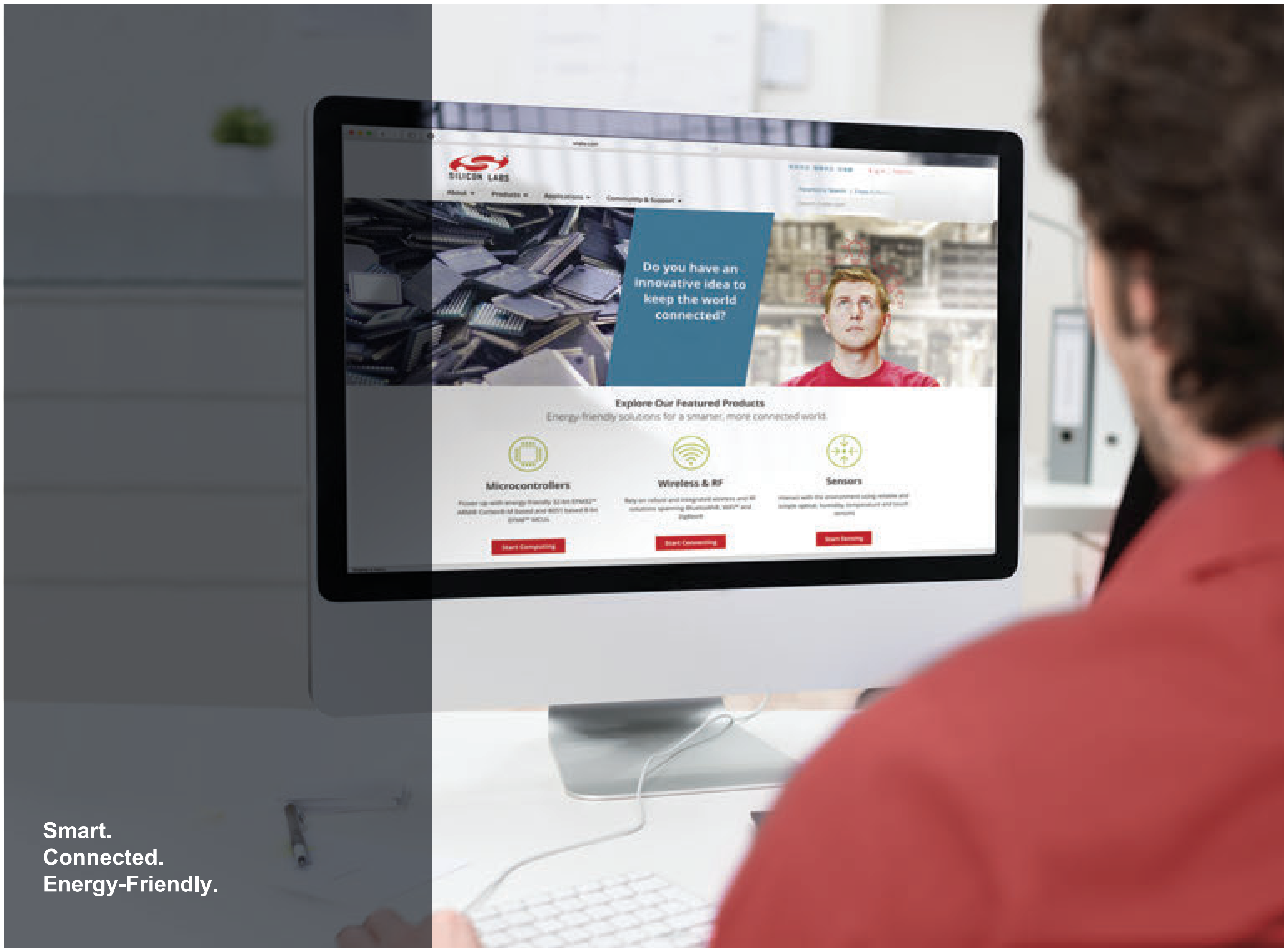
- **Security errors**

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



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com