



Bluetooth Smart Ready API Reference



This document contains the Silicon Labs Bluetooth Smart Ready Stack version unspecified API reference for the BGAPI serial protocol, BGLIB C API, and BGScript scripting language.

A short overview of the Silicon Labs Bluetooth Smart Ready Stack and SDK, and included tools, is also presented here.

Table of Contents

| | |
|---------------------------------------------------------------------------|-----------|
| 1. Silicon Labs Bluetooth Smart Ready Software | 4 |
| 1.1 Silicon Labs Bluetooth Smart Ready Stack | 5 |
| 1.2 Silicon Labs Bluetooth Smart Ready SDK | 6 |
| 1.3 BGAPI™ serial protocol API | 7 |
| 1.4 BGLIB™ Host Library. | 8 |
| 1.5 BGScript™ Scripting Language | 9 |
| 1.6 BGAPI™ vs BGScript™ | 11 |
| 1.7 Profile Toolkit™. | 12 |
| 1.8 BGBuild compiler | 12 |
| 1.9 DFU tools. | 12 |
| 1.10 BGTool | 13 |
| 2. Data types | 14 |
| 3. API Reference | 15 |
| 3.1 Connection management for Bluetooth Classic (bt_connection) | 16 |
| 3.1.1 bt_connection commands | 16 |
| 3.1.2 bt_connection events | 24 |
| 3.1.3 bt_connection enumerations | 30 |
| 3.2 Generic Access Profile, Bluetooth Classic (bt_gap) | 32 |
| 3.2.1 bt_gap commands | 32 |
| 3.2.2 bt_gap events | 49 |
| 3.2.3 bt_gap enumerations | 52 |
| 3.3 HID (bt_hid) | 53 |
| 3.3.1 bt_hid commands | 53 |
| 3.3.2 bt_hid events | 62 |
| 3.3.3 bt_hid enumerations. | 68 |
| 3.4 RFCOMM (bt_rfcomm) | 71 |
| 3.4.1 bt_rfcomm commands | 71 |
| 3.4.2 bt_rfcomm events | 80 |
| 3.5 Service Discovery Profile (bt_sdp) | 85 |
| 3.5.1 bt_sdp commands | 85 |
| 3.5.2 bt_sdp events | 89 |
| 3.6 Device Firmware Upgrade (dfu) | 90 |
| 3.6.1 dfu commands | 90 |
| 3.6.2 dfu events | 94 |
| 3.7 Endpoint (endpoint) | 96 |
| 3.7.1 endpoint commands. | 96 |
| 3.7.2 endpoint events | 102 |
| 3.7.3 endpoint enumerations | 107 |
| 3.7.4 endpoint defines | 108 |
| 3.8 Persistent Store (flash) | 109 |

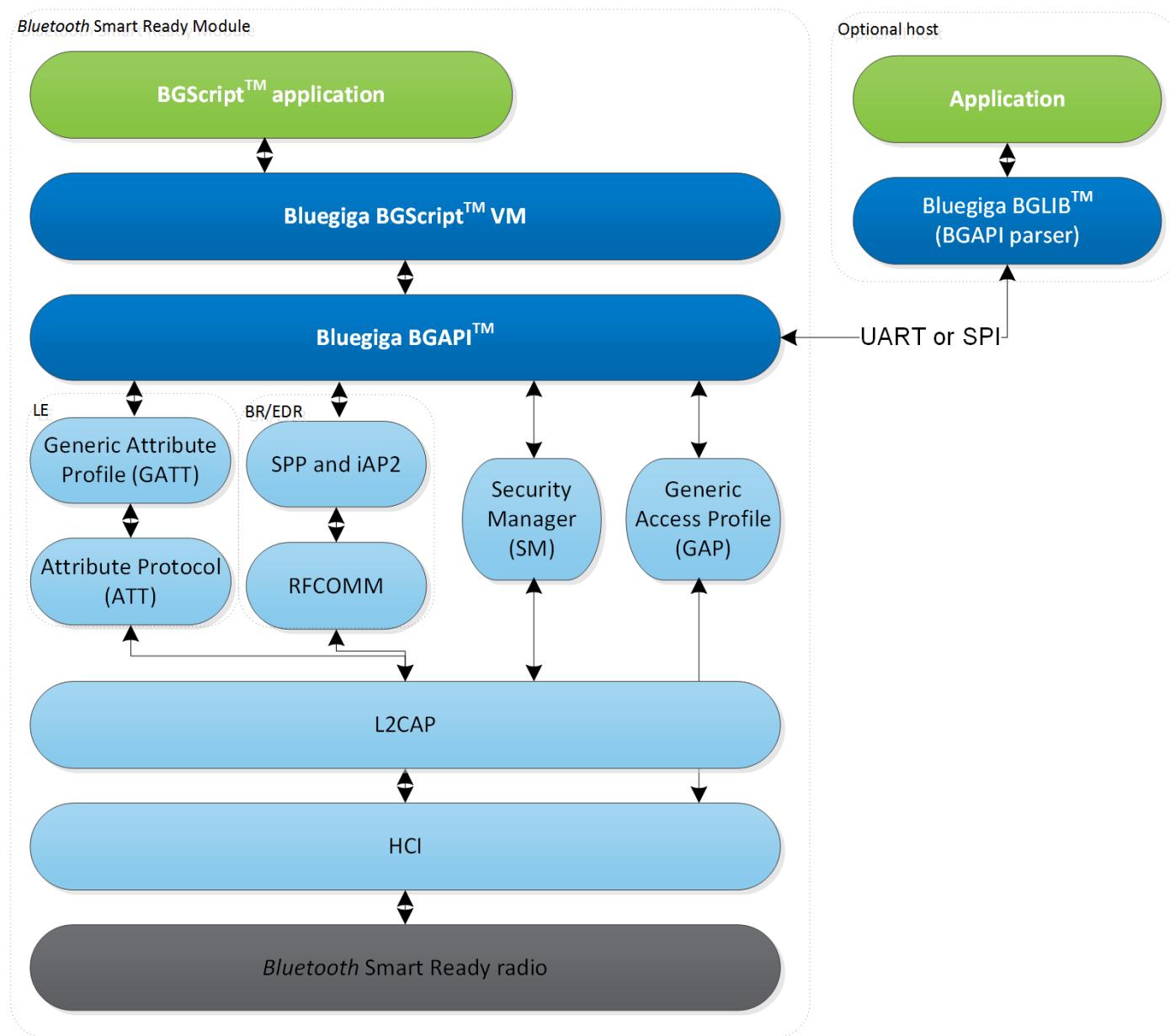
| | |
|-------------------------------------------------------------------------------|------------|
| 3.8.1 flash commands | 109 |
| 3.8.2 flash events | 114 |
| 3.9 Generic Attribute Profile (gatt) | 116 |
| 3.9.1 gatt commands | 116 |
| 3.9.2 gatt events | 147 |
| 3.9.3 gatt enumerations | 153 |
| 3.10 Generic Attribute Profile Server (gatt_server) | 155 |
| 3.10.1 gatt_server commands | 155 |
| 3.10.2 gatt_server events | 162 |
| 3.10.3 gatt_server enumerations | 166 |
| 3.11 Hardware (hardware) | 167 |
| 3.11.1 hardware commands | 167 |
| 3.11.2 hardware events | 182 |
| 3.11.3 hardware enumerations | 184 |
| 3.12 Identity Profile (identity) | 186 |
| 3.12.1 identity commands | 186 |
| 3.12.2 identity events | 189 |
| 3.13 Connection management for Bluetooth Low Energy (le_connection) | 191 |
| 3.13.1 le_connection commands | 191 |
| 3.13.2 le_connection events | 194 |
| 3.13.3 le_connection enumerations | 198 |
| 3.14 Generic Access Profile, Bluetooth Low Energy (le_gap) | 200 |
| 3.14.1 le_gap commands | 200 |
| 3.14.2 le_gap events | 216 |
| 3.14.3 le_gap enumerations | 218 |
| 3.15 LE Cable Replacement (le_serial) | 221 |
| 3.15.1 le_serial commands | 221 |
| 3.15.2 le_serial events | 225 |
| 3.16 Security Manager (sm) | 226 |
| 3.16.1 sm commands | 226 |
| 3.16.2 sm events | 240 |
| 3.16.3 sm enumerations | 248 |
| 3.17 System (system) | 249 |
| 3.17.1 system commands | 249 |
| 3.17.2 system events | 257 |
| 3.18 Testing commands (test) | 262 |
| 3.18.1 test commands | 262 |
| 3.18.2 test events | 271 |
| 3.18.3 test enumerations | 271 |
| 3.19 Utilities for BGScript (util) | 273 |
| 3.19.1 util commands | 273 |
| 3.20 Error codes. | 274 |
| 4. Document Revision History | 281 |

1. Silicon Labs Bluetooth Smart Ready Software

This section contains a short description of the Silicon Labs Bluetooth Smart Ready software, the components, APIs, and tools it includes.

1.1 Silicon Labs Bluetooth Smart Ready Stack

The main components of the Silicon Labs Bluetooth Smart Ready stack are shown in the figure below. The figure shows the layers of the Bluetooth Smart Ready stack as well as the APIs that can be used to interface to the Silicon Labs Bluetooth Smart Ready Stack.



Key features of the Silicon Labs Bluetooth Smart Ready stack include:

- Bluetooth 4.2 Smart Ready compatible
 - Bluetooth GAP and Security Manager
 - Bluetooth SPP and Apple iAP profiles
 - GATT over BR profile
 - Any GATT based Bluetooth Smart profile
- High performance features
 - 6 simultaneous BR/EDR connections
 - 7 simultaneous LE connections
 - 1 x BR/EDR and 6 x LE connections simultaneously
 - Up to 1 Mbps throughput over SPP
 - Up to 200 kbps throughput over iAP2

1.2 Silicon Labs Bluetooth Smart Ready SDK

The Silicon Labs Bluetooth Smart Ready SDK is a software development kit, which enables the device and software vendors to develop applications on top of the Silicon Labs's Bluetooth Smart Ready hardware and stack software.

The Bluetooth Smart Ready SDK supports multiple development models and the application developers can decide whether the application software runs on a separate host (a low power MCU) or whether they want to make fully standalone devices and execute their code on the MCU embedded in the Silicon Labs Bluetooth Smart Ready modules.

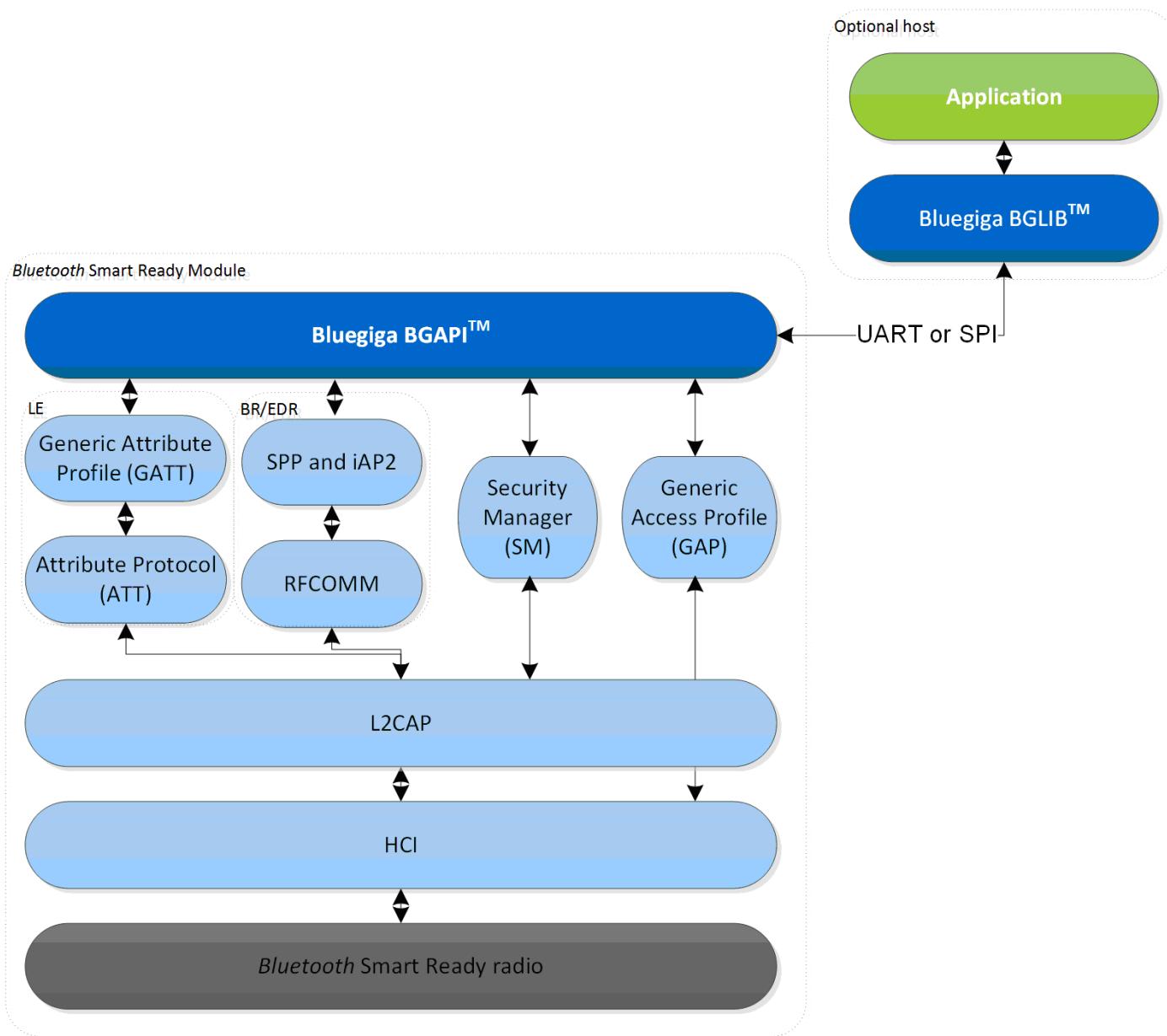
The SDK also contains documentation, tools for compiling the firmware, installing it into the hardware and lot of example applications speeding up the development process.

The Silicon Labs Bluetooth Smart Ready SDK includes the following APIs, components and tools:

- **The Bluetooth Smart Ready stack** as described in the previous chapter.
- **BGAPI™** is a binary serial protocol API to the Silicon Labs Bluetooth Smart Ready stack over UART or SPI interfaces. BGAPI is target for users, who want to use both Bluetooth BR/EDR and LE functionality and use all the features in the Bluetooth Smart Ready stack from an external host such as a low power MCU.
- **BGLIB™** is a host library for external MCUs and implements a reference parser for the BGAPI serial protocol. BGLIB is delivered in C source code as part of the Bluetooth Smart Ready SDK and it can be easily ported to various processor architectures.
- **BGScript™** interpreter and scripting language allow applications to be developed into the Bluetooth Smart Ready modules built-in MCU. It allows simple end user applications or enhanced functionality to be developed directly into the Bluetooth Smart Ready module which means no external host MCU is necessarily needed. BGScript applications can be executed at the same time as the BGAPI is used, allowing the possibility to implement some functionality on the Bluetooth module and some on the host.
- **Profile Toolkit™** is a simple XML based description language which can be used to easily and quickly develop GATT based service and characteristic databases for the Bluetooth Smart module.
- **BGBuild compiler** is a free-of-charge compiler that compiles the Bluetooth Smart Ready Stack, the BGScript application and the Bluetooth GATT services into the firmware binary that can be installed to the Bluetooth Smart Ready modules.
- **BGTool** is a graphical user interface application and a developer tool, which allows the Bluetooth smart module to be controller over the host interface using the BGAPI serial protocol. BGTool is a useful tool for testing the Bluetooth Smart module and evaluating its functionality and APIs.
- **DFU Tools** are also included as part of the SDK allowing the firmware to be updated over the UART and SPI interfaces.

1.3 BGAPI™ serial protocol API

The BGAPI serial protocol allows external hosts to interface to the Bluetooth Smart Ready modules over UART or SPI interfaces. The BGAPI serial protocol is a lightweight and well defined binary protocol which allows command and data to be sent to the Bluetooth Smart Ready module and it also provides a way to receive responses and events and data from the module.



The BGAPI provides access to the following layers in the *Bluetooth Smart Stack*:

- **BT GAP** - BT GAP provides access to basic Bluetooth BR/EDR features, like device discovery and connection establishment.
- **LE GAP** - LE GAP provides access to basic Bluetooth LE features, like device advertisement, discovery and connection establishment.
- **Security manager** - Security manager is used to configure the local devices security features and establish secure connections
- **RFCOMM** - RFCOMM provides basic serial data transmission over Bluetooth RD/EDR.
- **iAP** - iAP provides basic serial data transmission to Apple iOS devices using Bluetooth BR/EDR. iAP is only available to Apple MFI licenses and not included in the standard SDK.
- **Hardware** - Access to local hardware features and interfaces like SPI, I2C, GPIO and ADC
- **Persistent Store** - A data storage that allows data to be stored to and read from the internal flash
- **System** - Local device's status and management functions

1.4 BGLIB™ Host Library

The BGLIB host library is a reference implementation of the BGAPI serial protocol parser and it's provided in an ANSI C source code in the Bluetooth Smart Ready SDK. It abstracts the complexity of the BGAPI serial protocol and instead provides high level C functions and call-back handlers to the application developer, which makes the application development easier and faster.

The BGLIB library can be ported to various host systems ranging from low cost MCUs to devices running Linux, Windows or OSX.

```
/* Function to open RFCOMM connection */
void dumo_cmd_bt_rfcomm_open(    bd_addr address,
                                uint8 streaming_destination,
                                uint8 uuid_len,
                                const uint8* uuid_data);

/* Callback handler*/
struct dumo_msg_bt_rfcomm_open_rsp_t
{
    uint16 result,
    uint8 endpoint
}

/* Response handler */
void dumo_rsp_bt_rfcomm_open(
    const struct dumo_msg_bt_rfcomm_open_rsp_t *msg
)
```

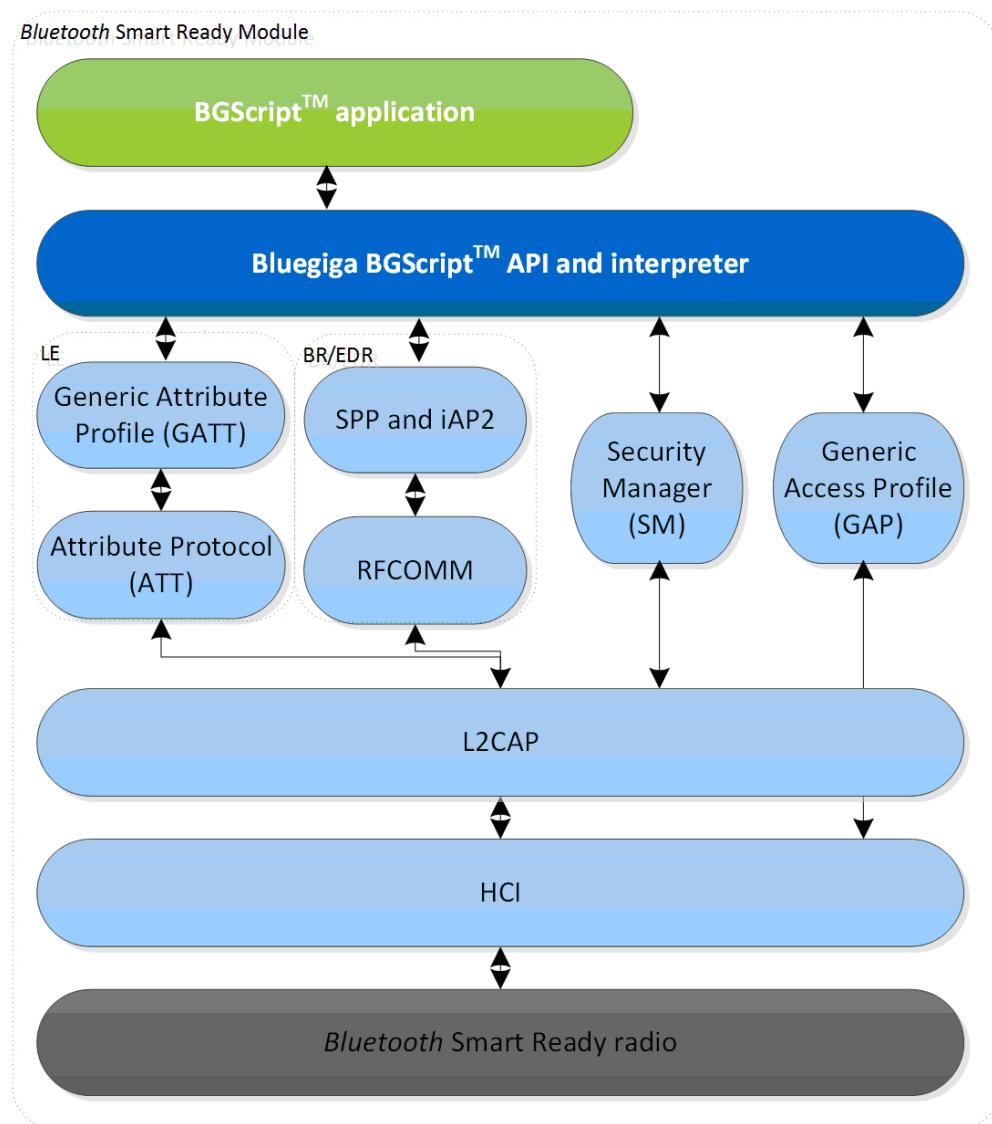
1.5 BGScript™ Scripting Language

BGScript is a simple BASIC-style programming language that allows end-user applications to be embedded to the Silicon Labs Bluetooth Smart Ready modules. Although being a simple and easy-to-learn programming language BGScript does provide features and functions to create fairly complex and powerful applications and it provides access to all the same APIs as the BGAPI serial protocol.

BGScript is fully event based programming language and code execution is started when events such as system start-up, Bluetooth connection, I/O interrupt etc. occur.

BGScript applications are developed with Silicon Labs's free-of-charge Bluetooth Smart Ready SDK and the BGScript applications are executed in the BGScript Virtual Machine (VM) that is part of the Silicon Labs Bluetooth Smart Ready software. The Bluetooth Smart Ready SDK comes with all the necessary tools for code editing and compilation and also the needed tools for installing the compiled firmware binary to the Silicon Labs Bluetooth Smart Ready modules.

BGScript applications can also be used at the same time as BGAPI and they can be for example used to automate some simple actions.



```
# System boot event listener - this is executed on module power on
event system_boot(major,minor,patch,build,bootloader,hw)

    # Start a software timer
    call hardware_set_soft_timer(1000,0,0)
end

# System initialized event - generated when the hardware is ready
# to be used
event system_initialized(addr)

    # Start RFCOMM server for SPP
    call bt_rfcomm_start_server(5,2,0)

    # Make device visible via both BT BR/EDR and LE
    call bt_gap_set_mode(1,1,0)
    call le_gap_set_le_mode(2,2)

    #Set local devices friendly name
    call system_set_local_name(21,"Bluegiga BT121 bgdemo")
end
```

1.6 BGAPI™ vs BGScript™

This section describes the differences between using BGAPI and BGScript. In brief the difference is:

- BGScript is our custom scripting language used for on-module applications. BGScript applications only run on Silicon Labs modules and dongles.
- BGAPI is a custom serial protocol used to externally control the modules over the host interface and BGLIB is an ANSI C reference implementation of the BGAPI serial protocol and only runs outside of our modules and dongles.

So the main difference between BGScript and BGLIB is that BGScript allows you to run an application right on the Bluetooth module, whereas BGLIB uses the BGAPI serial protocol API to send commands and receive events from an external device - typically a microcontroller. Note however that BGScript and BGLIB implement the exact same functionality. Every BGScript command, response, and event has a counterpart in the BGAPI serial protocol and BGLIB host library.

One other thing to keep in mind is that BGScript has some performance penalties compared to external API control due to the fact that BGScript is an interpreted scripting language and requires extra overhead in order to use. It makes the Bluetooth module do the work that could otherwise be done elsewhere. If you are trying to achieve maximum performance or you have an application which is fairly complex (lots of fast timers, interrupts, or communicating with many external sensors over I2C or SPI for example), it is often a good idea to use a small external microcontroller and BGLIB/BGAPI instead.

| Question | BGAPI™ | BGScript™ |
|-------------------------------------------------|----------------------------------|------------------------------------------------------------------------------------|
| An external host needed? | Yes | No |
| Host interface | UART or SPI** | No separate host needed |
| Bluetooth API | BGAPI or BGLIB APIs | BGScript API |
| Peripheral interface APIs and support | Host dependent (*) | APIs for UART, SPI, I2C, GPIO, ADC and PS store |
| Custom peripheral interface drivers | Can be developed to the host | Peripheral drivers are part of the Silicon Labs <i>Bluetooth</i> Smart Ready stack |
| RAM available for application | Host dependent | TBD |
| Flash available for application | Host dependent (**) | 0 - 24kB |
| Execution speed | Host dependent | TBD |
| Application development SDK | Host dependent + BGAPI and BGLIB | Silicon Labs <i>Bluetooth</i> Smart Ready SDK |
| <i>Bluetooth</i> firmware / application updates | DFU over UART or SPI** | DFU over UART or SPI** |

*) The *Bluetooth* Smart modules peripheral interfaces are still available via BGAPI commands and can be used to extend the host MCUs I/Os.

**) Check SPI host interface support status.

1.7 Profile Toolkit™

The *Bluetooth* Smart profile toolkit is a simple set of tools, which can be used to describe GATT based service and characteristic databases used with Bluetooth Smart and GATT over BR profiles. The profile toolkit consists of a simple XML based description language and templates, which can be used to describe the services and characteristics and their properties in a device's GATT database.

The GATT database developed with the Profile Toolkit is included as part of the device's firmware when the firmware is compiled.

```
<gatt>

<service uuid="1800">

    <description>Generic Access Service</description>

    <characteristic uuid="2a00">
        <properties read="true" const="true" />
        <value>BT121 Bluetooth SR</value>
    </characteristic>

    <characteristic uuid="2a01">
        <properties read="true" const="true" />
        <value type="hex">0</value>
    </characteristic>

</service>

</gatt>
```

1.8 BGBuild compiler

The BGBuild compiler is a simple compiler that is used to build firmware images for the Bluetooth Smart Ready modules. The BGBuild compiler compiles the Bluetooth Smart Ready stack, the GATT database and optionally also a BGScript application into a single firmware binary image that can be installed into a Bluetooth Smart module.

1.9 DFU tools

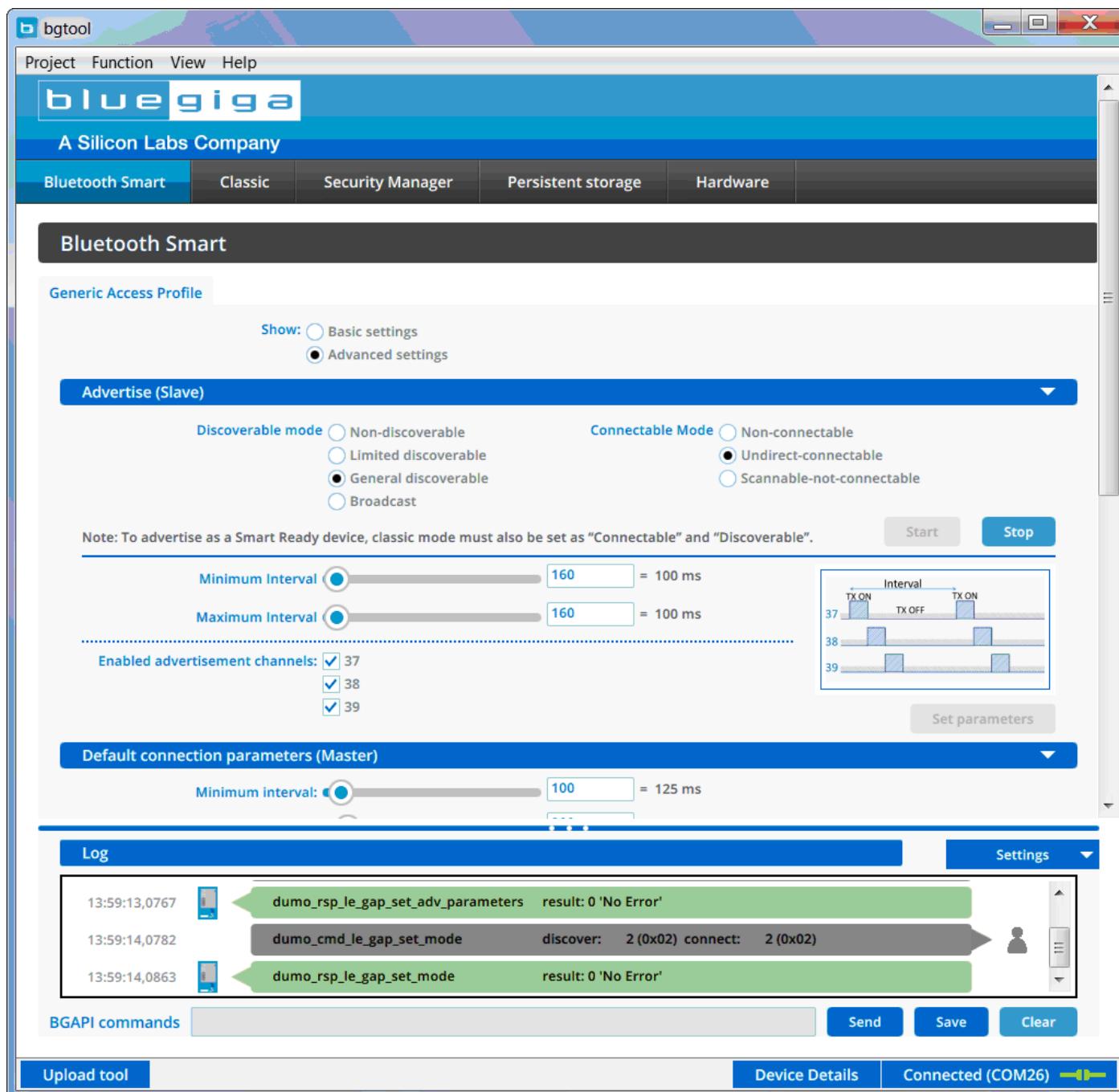
The Device Firmware Update (DFU) protocol is an open serial protocol that can be used to perform field updates to the Bluetooth Smart Ready modules. DFU protocol allows any firmware image generated with the BGBuild compiler to be installed into a Bluetooth Smart Module.

The Bluetooth Smart Ready SDK contains command line binaries versions and source code for the DFU protocol and tools.

DFU protocol and available commands are described in the API reference document.

1.10 BGTool

The BGTool application can be used to test and evaluate the Bluetooth Smart Ready module and issue BGAPI commands to it over the UART host interface.



2. 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 2.1. Data types

| Name | Length | BGScript equivalent | Description |
|----------------|---------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| errorcode | 2 bytes | Number | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| int16 | 2 bytes | Number | Signed 16-bit integer |
| bd_addr | 6 bytes | Array of 6 bytes | Bluetooth address |
| uint16 | 2 bytes | Number | Unsigned 16-bit integer |
| int32 | 4 bytes | Number | Signed 32-bit integer |
| uint32 | 4 bytes | Number* | Unsigned 32-bit integer |
| link_id_t | 2 bytes | Number | Link ID |
| int8 | 1 byte | Number | Signed 8-bit integer |
| uint8 | 1 byte | Number | Unsigned 8-bit integer |
| uint8array | 1 - 256 bytes | Array | Variable length byte array. First byte is the length of the array. |
| dbm | 1 byte | Number | Signal strength |
| connection | 1 byte | Number | Connection handle |
| service | 1 byte | Number | GATT service handle. This value is normally received from the gatt_service event. |
| characteristic | 1 byte | Number | GATT characteristic handle. This value is normally received from the gatt_characteristic event |
| descriptor | 1 byte | Number | The handle of the GATT characteristic descriptor |
| uuid | 1 byte | Number | Characteristic UUID, first byte is the length of UUID and rest is UUID in little-endian format |
| att_errorcode | 1 byte | Number | Attribute protocol error code. Values: <ul style="list-style-type: none"> • 0: No error • Non-zero: see error codes |
| att_opcode | 1 byte | Number | Attribute opcode which informs the procedure from which attribute the value was received |

(*) The script's internal number type is a signed 32-bit integer. This means large unsigned 32-bit integers in the BGAPI will be represented with negative numbers in the script. This is a known limitation.

3. API Reference

3.1 Connection management for Bluetooth Classic (bt_connection)

These commands and events are related to Bluetooth Classic connection management and provide the means to open, close and monitor Bluetooth Classic connections.

3.1.1 bt_connection commands

3.1.1.1 cmd_bt_connection_get_rssi

Get RSSI value of a connection.

Table 3.1. Command

| Byte | Type | Name | Description |
|------|-------|------------|------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | connection | BR/EDR Connection, LE Connection or RFCOMM handle |

Table 3.2. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 |

BGScript command

```
call bt_connection_get_rssi(connection)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_get_rssi(uint8 connection);

/* Response id */
dumo_rsp_bt_connection_get_rssi_id

/* Response structure */
struct dumo_msg_bt_connection_get_rssi_rsp_t
{
    uint16 result
}
```

Table 3.3. Events Generated

| Event | Description |
|------------------------------------------|-----------------------------|
| bt_connection_rssi_value | RSSI value of a connection. |

3.1.1.2 cmd_bt_connection_list

This command can be used to list all RFCOMM and HID connections and check parameters of each connection.

Table 3.4. Command

| Byte | Type | Name | Description |
|------|------|--------|------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x00 | method | Message ID |

Table 3.5. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 |

BGScript command

```
call bt_connection_list()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_list();

/* Response id */
dumo_rsp_bt_connection_list_id

/* Response structure */
struct dumo_msg_bt_connection_list_rsp_t
{
    uint16 result
}
```

Table 3.6. Events Generated

| Event | Description |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bt_connection_parameters | This event indicates the details of a single RFCOMM or HID connection. This event can be triggered when needed with bt_connection_list . For HID connections only fields address, powermode, role and encryption contain valid information. |
| bt_connection_list_complete | This event indicates that all connections have been listed. |

3.1.1.3 cmd_bt_connection_read_clock

Read Bluetooth Clock of a connection/piconet. For detailed information, see Bluetooth Specification 4.1, vol. 2, part E, section 7.5.6 (Read Clock Command).

Table 3.7. Command

| Byte | Type | Name | Description |
|------|-------|------------|----------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | connection | Connection endpoint handle or rfcomm handle. Zero to read local Bluetooth Clock. |

Table 3.8. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 |

BGScript command

```
call bt_connection_read_clock(connection)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_read_clock(uint8 connection);

/* Response id */
dumo_rsp_bt_connection_read_clock_id

/* Response structure */
struct dumo_msg_bt_connection_read_clock_rsp_t
{
    uint16 result
}
```

Table 3.9. Events Generated

| Event | Description |
|-------------------------------------------|------------------------------------------|
| bt_connection_clock_value | Bluetooth Clock of a connection/piconet. |

3.1.1.4 cmd_bt_connection_set_active

This command can be used to set the connection state to active.

Table 3.10. Command

| Byte | Type | Name | Description |
|------|-------|----------|------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | endpoint | Bluetooth connection endpoint handle |

Table 3.11. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 |

BGScript command

```
call bt_connection_set_active(endpoint)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_set_active(uint8 endpoint);

/* Response id */
dumo_rsp_bt_connection_set_active_id

/* Response structure */
struct dumo_msg_bt_connection_set_active_rsp_t
{
    uint16 result
}
```

Table 3.12. Events Generated

| Event | Description |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bt_connection_parameters | This event indicates the details of a single RFCOMM or HID connection. This event can be triggered when needed with bt_connection_list . For HID connections only fields address, powermode, role and encryption contain valid information. |

3.1.1.5 cmd_bt_connection_set_role

This command can be used to set the Bluetooth connection local role. Note that this may not be possible if the remote device is the master and does not allow role changes.

Table 3.13. Command

| Byte | Type | Name | Description |
|------|-------|----------|------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle |
| 5 | uint8 | role | Bluetooth connection local role |

Table 3.14. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 |

BGScript command

```
call bt_connection_set_role(endpoint,role)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_set_role(uint8 endpoint, uint8 role);

/* Response id */
dumo_rsp_bt_connection_set_role_id

/* Response structure */
struct dumo_msg_bt_connection_set_role_rsp_t
{
    uint16 result
}
```

Table 3.15. Events Generated

| Event | Description |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bt_connection_parameters | This event indicates the details of a single RFCOMM or HID connection. This event can be triggered when needed with <code>bt_connection_list</code> . For HID connections only fields address, powermode, role and encryption contain valid information. |

3.1.1.6 cmd_bt_connection_set_sniff

This command can be used to set the sniff parameters for a connection. Please see BLUETOOTH SPECIFICATION Version 4.1 [Vol 2] Part E chapter 7.2.2 for more information regarding the use of the related parameters.

Table 3.16. Command

| Byte | Type | Name | Description |
|-------|--------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x09 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | endpoint | Bluetooth connection endpoint handle |
| 5-6 | uint16 | max | Maximum sniff interval. <ul style="list-style-type: none"> • Range: 0x0004 to 0xffff • Only even values are valid • Sniff interval = 0.625ms x max • Time range 2.5 ms to 40959 ms |
| 7-8 | uint16 | min | Minimum sniff interval. <ul style="list-style-type: none"> • Range: 0x0002 to 0xffff • Only even values are valid • Sniff interval = 0.625ms x min • Time range 1.25 ms to 40959 ms |
| 9-10 | uint16 | attempt | Number of baseband receive slots for sniff attempt. <ul style="list-style-type: none"> • Range: 0x0001 to 0x7fff • Time range 0.625 ms to 40959 ms |
| 11-12 | uint16 | timeout | Number of baseband receive slots for sniff timeout. <ul style="list-style-type: none"> • Range: 0x0000 to 0x7fff • Time range 0.0 ms to 40959 ms |

Table 3.17. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_connection_set_sniff(endpoint,max,min,attempt,timeout)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_set_sniff(uint8 endpoint, uint16 max, uint16 min, uint16 attempt, uint16 timeout);

/* Response id */
dumo_rsp_bt_connection_set_sniff_id

/* Response structure */
struct dumo_msg_bt_connection_set_sniff_rsp_t
{
    uint16 result
}
```

Table 3.18. Events Generated

| Event | Description |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bt_connection_parameters | This event indicates the details of a single RFCOMM or HID connection. This event can be triggered when needed with bt_connection_list . For HID connections only fields address, powermode, role and encryption contain valid information. |

3.1.1.7 cmd_bt_connection_set_supervision_timeout

This command can be used to set the connection supervision timeout.

Table 3.19. Command

| Byte | Type | Name | Description |
|------|--------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | endpoint | Bluetooth connection endpoint handle |
| 5-6 | uint16 | supervisiontimeout | Supervision timeout defines how long, in multiples of 0.625 ms, the connection can be without any activity before being disconnected. Note that this command has no effect if the local device is not the Master of the connection. Range: 0x0001 - 0xffff. Default value: 0x7D00 (20 seconds) |

Table 3.20. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 |

BGScript command

```
call bt_connection_set_supervision_timeout(endpoint,supervisiontimeout)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_connection_set_supervision_timeout(uint8 endpoint, uint16 supervisiontimeout);

/* Response id */
dumo_rsp_bt_connection_set_supervision_timeout_id

/* Response structure */
struct dumo_msg_bt_connection_set_supervision_timeout_rsp_t
{
    uint16 result
}
```

3.1.2 bt_connection events

3.1.2.1 evt_bt_connection_clock_value

Bluetooth Clock of a connection/piconet.

Table 3.21. Event

| Byte | Type | Name | Description |
|------|--------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | connection | Connection endpoint handle or zero when reading local clock. NOTE: this is always a connection handle regardless whether clock was requested by RFCOMM or connection handle. |
| 5-8 | uint32 | clock | Bluetooth Clock of the device requested. |
| 9-10 | uint16 | accuracy | Maximum Bluetooth Clock error in +- 0.3125 ms units. 0xFFFF means unknown. |

BGScript event

```
event bt_connection_clock_value(connection,clock,accuracy)
```

C Functions

```
/* Event id */
dumo_evt_bt_connection_clock_value_id

/* Event structure */
struct dumo_msg_bt_connection_clock_value_evt_t
{
    uint8 connection,
    uint32 clock,
    uint16 accuracy
}
```

3.1.2.2 evt_bt_connection_closed

This event indicates that a connection was closed.

Note: This event corresponds to the Bluetooth connection (not the endpoint) between two devices closing. Once all open endpoints have closed, either side may close the connection. Thus the reason parameter may be either remote_user_terminated or connection_terminated_by_local_host independent of which side closed the last endpoint.

Table 3.22. Event

| Byte | Type | Name | Description |
|------|--------|------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 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 | Bluetooth connection handle |

BGScript event

```
event bt_connection_closed(reason,connection)
```

C Functions

```
/* Event id */
dumo_evt_bt_connection_closed_id

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

3.1.2.3 evt_bt_connection_list_complete

This event indicates that all connections have been listed.

Table 3.23. Event

| Byte | Type | Name | Description |
|------|------|--------|------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x03 | method | Message ID |

BGScript event

```
event bt_connection_list_complete()
```

C Functions

```
/* Event id */
dumo_evt_bt_connection_list_complete_id

/* Event structure */
struct dumo_msg_bt_connection_list_complete_evt_t
{
}
```

3.1.2.4 evt_bt_connection_opened

This event indicates that a new connection was opened.

Table 3.24. Event

| Byte | Type | Name | Description |
|------|---------|------------|----------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of the remote device in little endian formats |
| 10 | uint8 | master | Bluetooth role of the local device. Values: • 0: Slave • 1: Master |
| 11 | uint8 | connection | Bluetooth connection handle |
| 12 | uint8 | bonding | Bonding handle of the remote device • 0xff: No bonding • Other: Bonding handle |

BGScript event

```
event bt_connection_opened(address,master,connection,bonding)
```

C Functions

```
/* Event id */
dumo_evt_bt_connection_opened_id

/* Event structure */
struct dumo_msg_bt_connection_opened_evt_t
{
    bd_addr *address,
    uint8 master,
    uint8 connection,
    uint8 bonding
}
```

3.1.2.5 evt_bt_connection_parameters

This event indicates the details of a single RFCOMM or HID connection. This event can be triggered when needed with [bt_connection_list](#). For HID connections only fields address, powermode, role and encryption contain valid information.

Table 3.25. Event

| Byte | Type | Name | Description |
|-------|---------|--------------|----------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x15 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | endpoint | RFCOMM connection endpoint handle |
| 5-8 | uint32 | block_size | Size of a single packet over RFCOMM |
| 9 | uint8 | msc | Bit mask for MSC at the local device |
| 10-15 | bd_addr | address | Bluetooth address of the remote device in little endian format |
| 16 | uint8 | direction | Direction of the connection |
| 17 | uint8 | powermode | Power mode of the connection |
| 18 | uint8 | role | Role in the connection |
| 19 | uint8 | encryption | |
| 20-23 | uint32 | input_buffer | Amount of data in the RFCOMM receive buffer |
| 24 | uint8 | port | RFCOMM local port number that was connected. |

BGScript event

```
event
bt_connection_parameters(endpoint,block_size,msc,address,direction,powermode,role,encryption,input_buffer,port)
```

C Functions

```
/* Event id */
dumo_evt_bt_connection_parameters_id

/* Event structure */
struct dumo_msg_bt_connection_parameters_evt_t
{
    uint8 endpoint,
    uint32 block_size,
    uint8 msc,
    bd_addr *address,
    uint8 direction,
    uint8 powermode,
    uint8 role,
    uint8 encryption,
    uint32 input_buffer,
    uint8 port
}
```

3.1.2.6 evt_bt_connection_rssi_value

RSSI value of a connection.

Table 3.26. Event

| Byte | Type | Name | Description |
|------|-------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x07 | class | Message class: Connection management for Bluetooth Classic |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle. NOTE: this is always connection handle regardless whether RSSI was requested by RFCOMM or connection handle |
| 5 | int8 | rssi | For BR/EDR: Received signal strength in dB relative to 'Golden Receive Power Range' which is from -40dBm to -60dBm. Negative values indicate we are below minimum of the range and positive values indicate that we are above the maximum of the range For LE: Absolute power level in dBm to +/-6dB accuracy. If the RSSI cannot be read, the RSSI metric shall be set to 127. |

BGScript event

```
event bt_connection_rssi_value(connection,rssi)
```

C Functions

```
/* Event id */
dumo_evt_bt_connection_rssi_value_id

/* Event structure */
struct dumo_msg_bt_connection_rssi_value_evt_t
{
    uint8 connection,
    int8 rssi
}
```

3.1.3 bt_connection enumerations

3.1.3.1 enum_bt_connection_direction

These values indicate the direction of the connection.

Table 3.27. Enumerations

| Value | Name | Description |
|-------|------------------------|-------------|
| 0 | bt_connection_outgoing | Outgoing |
| 1 | bt_connection_incoming | Incoming |

3.1.3.2 enum_bt_connection_encryption

These values indicate the encryption status of the connection.

Table 3.28. Enumerations

| Value | Name | Description |
|-------|-------------------------|-------------|
| 0 | bt_connection_plain | Unencrypted |
| 1 | bt_connection_encrypted | Encrypted |

3.1.3.3 enum_bt_connection_powermode

These values indicate the power mode of a connection.

Table 3.29. Enumerations

| Value | Name | Description |
|-------|----------------------|-------------|
| 0 | bt_connection_active | Active |
| 1 | bt_connection_hold | Reserved |
| 2 | bt_connection_sniff | Sniff |
| 3 | bt_connection_park | Reserved |

3.1.3.4 enum_bt_connection_role

These values indicate the connection role of the local device.

Table 3.30. Enumerations

| Value | Name | Description |
|-------|----------------------|-------------|
| 0 | bt_connection_master | Master |
| 1 | bt_connection_slave | Slave |

3.2 Generic Access Profile, Bluetooth Classic (bt_gap)

These commands and events are related to Generic Access Profile, Bluetooth Classic.

3.2.1 bt_gap commands

3.2.1.1 cmd_bt_gap_cancel_discovery

This command can be used to cancel the ongoing Bluetooth Classic device discovery (inquiry) procedure.

Table 3.31. Command

| Byte | Type | Name | Description |
|------|------|--------|----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x01 | method | Message ID |

Table 3.32. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 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 |

BGScript command

```
call bt_gap_cancel_discovery()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_cancel_discovery();

/* Response id */
dumo_rsp_bt_gap_cancel_discovery_id

/* Response structure */
struct dumo_msg_bt_gap_cancel_discovery_rsp_t
{
    uint16 result
}
```

Table 3.33. Events Generated

| Event | Description |
|-------------------------------------------|---------------------------------------------------------|
| bt_gap_discovery_complete | This event indicates that discovery has been completed. |

3.2.1.2 cmd_bt_gap_discover

This command can be used to discover other Bluetooth Classic devices with Bluetooth Classic inquiry. Command [set_discovery_mode](#) can be used to select how much information is delivered in discovery results

Table 3.34. Command

| Byte | Type | Name | Description |
|------|-------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | timeout | The maximum amount of time (in units of 1.28 seconds) before the inquiry process is halted. Range: 1 to 48. <ul style="list-style-type: none"> Example: Value 5 corresponds to 5×1.28 seconds = 6.4 seconds. |
| 5-8 | int32 | lap | Flag which selects whether to seek only devices that are in general discovery mode or in limited discovery mode. |

Table 3.35. Response

| Byte | Type | Name | Description |
|------|--------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> 0: success Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_gap_discover(timeout,lap)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_discover(uint8 timeout, int32 lap);

/* Response id */
dumo_rsp_bt_gap_discover_id

/* Response structure */
struct dumo_msg_bt_gap_discover_rsp_t
{
    uint16 result
}
```

Table 3.36. Events Generated

| Event | Description |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bt_gap_discovery_result | This event returns the discovery result for a single remote device. The name parameter is present only when using extended discovery mode. The RSSI value is valid only when using RSSI or extended discovery (inquiry) mode. The discovery mode is set by set_discovery_mode command. |
| bt_gap_discovery_complete | This event indicates that discovery has been completed. |

3.2.1.3 cmd_bt_gap_get_mode

This command can be used to read the device's Bluetooth Classic visibility and connectability settings.

Table 3.37. Command

| Byte | Type | Name | Description |
|------|------|--------|----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x04 | method | Message ID |

Table 3.38. Response

| Byte | Type | Name | Description |
|------|--------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | connectable | Informs whether the device accepts incoming connections or not. Values: <ul style="list-style-type: none"> • 0: Not connectable • 1: Connectable |
| 7 | uint8 | discoverable | Informs whether the device is visible for inquiry. Values: <ul style="list-style-type: none"> • 0: Not discoverable • 1: Discoverable |
| 8 | uint8 | limited | Informs whether the device is visible only in limited mode inquiry or in general mode inquiry |

BGScript command

```
call bt_gap_get_mode()(result,connectable,discoverable,limited)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_get_mode();

/* Response id */
dumo_rsp_bt_gap_get_mode_id

/* Response structure */
struct dumo_msg_bt_gap_get_mode_rsp_t
{
    uint16 result,
    uint8 connectable,
    uint8 discoverable,
```

```
    uint8 limited
}
```

3.2.1.4 cmd_bt_gap_get_remote_name

This command can be used to read the name of the remote Bluetooth Classic device.

Table 3.39. Command

| Byte | Type | Name | Description |
|------|---------|---------|----------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x06 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x02 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of the remote Bluetooth Classic device in little endian format |

Table 3.40. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 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 |

BGScript command

```
call bt_gap_get_remote_name(address)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_get_remote_name(bd_addr *address);

/* Response id */
dumo_rsp_bt_gap_get_remote_name_id

/* Response structure */
struct dumo_msg_bt_gap_get_remote_name_rsp_t
{
    uint16 result
}
```

Table 3.41. Events Generated

| Event | Description |
|--------------------|--------------------------------------------------------|
| bt_gap_remote_name | This event indicates a reply to a remote name request. |

3.2.1.5 cmd_bt_gap_open

This command can be used to open a Bluetooth Classic connection to a remote device. After connection is established, closing another Bluetooth Classic endpoints (like RFCOMM or HID) will not close this endpoint. It is needed to be closed manually with [dumo_cmd_endpoint_close](#). Command can be used for example when there is a need to bond the devices without opening an RFCOMM connection. After connection has been opened bonding can be done with the [sm_increase_security](#) command.

Table 3.42. Command

| Byte | Type | Name | Description |
|------|---------|---------|----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x08 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address |

Table 3.43. Response

| Byte | Type | Name | Description |
|------|--------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | connection | Unique connection handle |

BGScript command

```
call bt_gap_open(address)(result,connection)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_open(bd_addr *address);

/* Response id */
dumo_rsp_bt_gap_open_id

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

Table 3.44. Events Generated

| Event | Description |
|----------------------|-------------|
| bt_connection_opened | |

3.2.1.6 cmd_bt_gap_set_auto_sniff

Set automatic sniff parameters for all connections. Please see the BLUETOOTH SPECIFICATION Version 4.1 [Vol 2] Part E chapter 7.2.2 for more information regarding the use of the related parameters.

Table 3.45. Command

| Byte | Type | Name | Description |
|-------|--------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | mode | Defines automatic sniff mode. Values: <ul style="list-style-type: none">• 0: Do not use automatic sniff• 1: Set connection active whenever data is transmitted• 2: Set connection to sniff when idle time is exceeded• 3: Set both active and sniff automatically |
| 5-6 | uint16 | idle | The time (in seconds) a link has to be idle before it will be set to sniff mode. Range: 1 - 250 |
| 7-8 | uint16 | max | Maximum sniff interval. <ul style="list-style-type: none">• Range: 0x0004 - 0xffffe• Only even values are valid• Sniff interval = 0.625ms x max• Time range 2.5 ms to 40959 ms Recommended default: 0x00a0 - 0x0640 (100 ms - 1000 ms) |
| 9-10 | uint16 | min | Minimum sniff interval. <ul style="list-style-type: none">• Range: 0x0002 - 0xffffe• Only even values are valid• Sniff interval = 0.625ms x min• Time range 1.25 ms - 40959 ms Recommended default: 0x20 |
| 11-12 | uint16 | attempt | Number of baseband receive slots for sniff attempt. <ul style="list-style-type: none">• Range: 0x0001-0x7fff• Time range 0.625 ms to 40959 ms Recommended default: 1 |
| 13-14 | uint16 | timeout | Number of baseband receive slots for sniff timeout. <ul style="list-style-type: none">• Range: 0x0000-0x7fff• Time range 0.0 ms to 40959 ms Recommended default: 8 |

Table 3.46. Response

| Byte | Type | Name | Description |
|------|------|--------|----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x09 | method | Message ID |

| Byte | Type | Name | Description |
|------|--------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4-5 | uint16 | result | <p>Result code</p> <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred <p>For other values refer to the Error codes</p> |

BGScript command

```
call bt_gap_set_auto_sniff(mode,idle,max,min,attempt,timeout)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_auto_sniff(uint8 mode, uint16 idle, uint16 max, uint16 min, uint16 attempt, uint16 timeout);

/* Response id */
dumo_rsp_bt_gap_set_auto_sniff_id

/* Response structure */
struct dumo_msg_bt_gap_set_auto_sniff_rsp_t
{
    uint16 result
}
```

3.2.1.7 cmd_bt_gap_set_discovery_mode

This command can be used to set the discovery mode. This determines the information given in [discovery_result](#) events.

Table 3.47. Command

| Byte | Type | Name | Description |
|------|-------|--------|---------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0xa0 | method | Message ID |
| 4 | uint8 | mode | Format on discovery results: 0: Standard short format. 1: With RSSI result. 2: Extended format. Value at startup: 1 |

Table 3.48. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0xa0 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_gap_set_discovery_mode(mode)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_discovery_mode(uint8 mode);

/* Response id */
dumo_rsp_bt_gap_set_discovery_mode_id

/* Response structure */
struct dumo_msg_bt_gap_set_discovery_mode_rsp_t
{
    uint16 result
}
```

3.2.1.8 cmd_bt_gap_set_host_channel_classification

This command configures Bluetooth Classic channel classifications.. If successful, a [bt_gap_host_channel_classification_complete](#) event will follow when the controller has completed the configuration.

Table 3.49. Command

| Byte | Type | Name | Description |
|------|------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x0b | method | Message ID |
| 4 | uint8array | channel_map | Channel bitmap for 79 channels. Mark known bad channels with a zero bit, other channels with a one bit. The parameter must be given as exactly 10 bytes; the highest bit is ignored. |

Table 3.50. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 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 |

BGScript command

```
call bt_gap_set_host_channel_classification(channel_map_len, channel_map_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_host_channel_classification(uint8array channel_map);

/* Response id */
dumo_rsp_bt_gap_set_host_channel_classification_id

/* Response structure */
struct dumo_msg_bt_gap_set_host_channel_classification_rsp_t
{
    uint16 result
}
```

3.2.1.9 cmd_bt_gap_set_max_power

This command can be used to set the maximum TX power for Bluetooth Classic.

Table 3.51. Command

| Byte | Type | Name | Description |
|------|------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x07 | method | Message ID |
| 4 | int8 | max_power | Maximum radio TX power. <ul style="list-style-type: none"> Value range: -20 - 12. Default value: 12 Note that the power measured at the antenna is slightly (1-3 dB) lower than this value. |

Table 3.52. Response

| Byte | Type | Name | Description |
|------|--------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> 0: success Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_gap_set_max_power(max_power)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_max_power(int8 max_power);

/* Response id */
dumo_rsp_bt_gap_set_max_power_id

/* Response structure */
struct dumo_msg_bt_gap_set_max_power_rsp_t
{
    uint16 result
}
```

3.2.1.10 cmd_bt_gap_set_mode

This command can be used to set Bluetooth Classic visibility and connectability.

Table 3.53. Command

| Byte | Type | Name | Description |
|------|-------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | connectable | Determines whether the device accepts incoming connections or not. Values: <ul style="list-style-type: none">• 0: Not connectable• 1: Connectable Default value: 0 |
| 5 | uint8 | discoverable | Determines whether the device is visible in Bluetooth Classic inquiry. Values: <ul style="list-style-type: none">• 0: Not discoverable• 1: Discoverable Default value: 0 |
| 6 | uint8 | limited | Determines whether the device is visible in limited mode inquiry or in general mode inquiry. Default value: general mode |

Table 3.54. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_gap_set_mode(connectable,discoverable,limited)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_mode(uint8 connectable, uint8 discoverable, uint8 limited);

/* Response id */
dumo_rsp_bt_gap_set_mode_id

/* Response structure */
struct dumo_msg_bt_gap_set_mode_rsp_t
{
```

```
    uint16 result  
}
```

3.2.1.11 cmd_bt_gap_set_parameters

This command can be used to set the default page timeout and scan mode.

Table 3.55. Command

| Byte | Type | Name | Description |
|------|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | pagetimeout | Page timeout defines how long the connection establishment can take before an error occurs, in multiples of 0.625 milliseconds. Range: 0x0001 - 0xffff. Default value: 0x2000 (5.12 seconds) |
| 6-7 | uint16 | scan_interval | Page scan interval in multiples of 0.625ms, values from 0x12 to 0x1000, only even numbers are allowed, see BT Core spec Vol. 2 part D chapter 7.3. Default value: 0x800 |
| 8-9 | uint16 | scan_window | Page scan window in multiples of 0.625ms, values from 0x11 to 0x1000, must be less than or equal to page scan interval. See BT Core spec Vol. 2 part D chapter 7.3. Default value: 0x12 |

Table 3.56. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_gap_set_parameters(pagetimeout,scan_interval,scan_window)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_parameters(uint16 pagetimeout, uint16 scan_interval, uint16 scan_window);

/* Response id */
dumo_rsp_bt_gap_set_parameters_id

/* Response structure */
struct dumo_msg_bt_gap_set_parameters_rsp_t
{
    uint16 result
}
```

3.2.1.12 cmd_bt_gap_set_policy

This command can be used to set default policies for connections. For more information on the use of policies please see BLUETOOTH SPECIFICATION Version 4.1 [Vol 2] Part E chapter 7.2.10.

Table 3.57. Command

| Byte | Type | Name | Description |
|------|-------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | allow_role_change | Defines if the local device accepts role change requests or not. Values: <ul style="list-style-type: none">• 0: Do not allow role change• 1: Allow role change Default value: 0 |
| 5 | uint8 | allow_sniff | This flag defines whether sniff mode is allowed or not. Values: <ul style="list-style-type: none">• 0: Do not allow sniff mode• 1: Allow sniff mode Default value: 0 |

Table 3.58. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_gap_set_policy(allow_role_change,allow_sniff)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_gap_set_policy(uint8 allow_role_change, uint8 allow_sniff);

/* Response id */
dumo_rsp_bt_gap_set_policy_id

/* Response structure */
struct dumo_msg_bt_gap_set_policy_rsp_t
{
    uint16 result
}
```

3.2.2 bt_gap events

3.2.2.1 evt_bt_gap_discovery_complete

This event indicates that discovery has been completed.

Table 3.59. Event

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | status | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript event

```
event bt_gap_discovery_complete(status)
```

C Functions

```
/* Event id */
dumo_evt_bt_gap_discovery_complete_id

/* Event structure */
struct dumo_msg_bt_gap_discovery_complete_evt_t
{
    uint16 status
}
```

3.2.2.2 evt_bt_gap_discovery_result

This event returns the discovery result for a single remote device. The name parameter is present only when using extended discovery mode. The RSSI value is valid only when using RSSI or extended discovery (inquiry) mode. The discovery mode is set by [set_discovery_mode](#) command.

Table 3.60. Event

| Byte | Type | Name | Description |
|-------|------------|---------------------------|---------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x0e | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | bd_addr | Bluetooth address of the discovered Bluetooth Classic device in little endian format |
| 10 | uint8 | page_scan_repetition_mode | Remote devices page scan repetition mode |
| 11-14 | uint32 | class_of_device | Class of Device for the device |
| 15 | int8 | rssi | RSSI value of the connection. • Range: -127 to +20 • Units: dBm |
| 16 | uint8 | bonding | Remote device's bonding handle. Values: • 0xff : Device is not bonded • Others : Bonding handle |
| 17 | uint8array | name | Remote device's Bluetooth Classic friendly name parsed from the Extended Inquiry Response (EIR) data |

BGScript event

```
event      bt_gap_discovery_result(bd_addr,page_scan_repetition_mode,class_of_device,rssi,bonding,name_len,
name_data)
```

C Functions

```
/* Event id */
dumo_evt_bt_gap_discovery_result_id

/* Event structure */
struct dumo_msg_bt_gap_discovery_result_evt_t
{
    bd_addr *bd_addr,
    uint8 page_scan_repetition_mode,
    uint32 class_of_device,
    int8 rssi,
    uint8 bonding,
    uint8array name
}
```

3.2.2.3 evt_bt_gap_host_channel_classification_complete

This event indicates host channel classification request is completed.

Table 3.61. Event

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | status | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript event

```
event bt_gap_host_channel_classification_complete(status)
```

C Functions

```
/* Event id */
dumo_evt_bt_gap_host_channel_classification_complete_id

/* Event structure */
struct dumo_msg_bt_gap_host_channel_classification_complete_evt_t
{
    uint16 status
}
```

3.2.2.4 evt_bt_gap_remote_name

This event indicates a reply to a remote name request.

Table 3.62. Event

| Byte | Type | Name | Description |
|------|------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x09 | olen | Minimum payload length |
| 2 | 0x02 | class | Message class: Generic Access Profile, Bluetooth Classic |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | status | Result code • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6-11 | bd_addr | address | Bluetooth address of the remote Bluetooth Classic device in little endian format |
| 12 | uint8array | remote_name | Name of the remote Bluetooth Classic device |

BGScript event

```
event bt_gap_remote_name(status,address,remote_name_len, remote_name_data)
```

C Functions

```
/* Event id */
dumo_evt_bt_gap_remote_name_id

/* Event structure */
struct dumo_msg_bt_gap_remote_name_evt_t
{
    uint16 status,
    bd_addr *address,
    uint8array remote_name
}
```

3.2.3 bt_gap enumerations

3.2.3.1 enum_bt_gap_discover_mode

These values define the GAP Discoverable mode of the module.

Table 3.63. Enumerations

| Value | Name | Description |
|-------|-------------------------|---------------------------------------------------------------|
| 0 | bt_gap_discover_generic | Discover devices which are in general discoverable mode. |
| 1 | bt_gap_discover_limited | Discover only devices which are in limited discoverable mode. |

3.3 HID (bt_hid)

HID connections Please note that HID commands are available only in images compiled with HID sdk

3.3.1 bt_hid commands

3.3.1.1 cmd_bt_hid_get_report_response

This command is used to respond to evt_bt_hid_get_report requests. The parameters endpoint, parameters_used, report_type, and report_id should match those received in the request. The report_data should contain the current instantaneous state of the fields specified in the request.

Table 3.64. Command

| Byte | Type | Name | Description |
|------|------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the request. |
| 5 | uint8 | parameters_used | |
| 6 | uint8 | report_type | |
| 7 | uint8 | report_id | ID of the report requested. Must match both report_type and ID, as the same ID can be used for different report types. |
| 8 | uint8array | report_data | The payload of the Report requested. Note that if the Report is larger than the max_bytes specified in the request, the report data must be truncated to max_bytes, if no report ID was requested; or max_bytes - 1 if an ID was requested. |

Table 3.65. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call      bt_hid_get_report_response(endpoint,parameters_used,report_type,report_id,report_data_len,
report_data_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_get_report_response(uint8 endpoint, uint8 parameters_used, uint8 report_type, uint8
report_id, uint8array report_data);

/* Response id */
dumo_rsp_bt_hid_get_report_response_id

/* Response structure */
struct dumo_msg_bt_hid_get_report_response_rsp_t
{
```

```
    uint16 result
}
```

3.3.1.2 cmd_bt_hid_get_set_report_reject

This command is used to reject an evt_bt_hid_get_report request for an invalid Report.

Table 3.66. Command

| Byte | Type | Name | Description |
|------|-------|----------|----------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the request. |
| 5 | uint8 | reason | Reason for rejecting. 0 = invalid Report ID, 1 = invalid Report Type or size mismatch. |

Table 3.67. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_hid_get_set_report_reject(endpoint,reason)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_get_set_report_reject(uint8 endpoint, uint8 reason);

/* Response id */
dumo_rsp_bt_hid_get_set_report_reject_id

/* Response structure */
struct dumo_msg_bt_hid_get_set_report_reject_rsp_t
{
    uint16 result
}
```

3.3.1.3 cmd_bt_hid_open

Open HID connection to remote HID Host. Note that the HID Host must have initiated a previous connection and stored our HID SDP entry, and our SDP entry must indicate support for HID Device -initiated connections; otherwise the HID Host will reject our connection attempt.

Table 3.68. Command

| Byte | Type | Name | Description |
|------|---------|-----------------------|-----------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth device address of the remote device in little endian format |
| 10 | uint8 | streaming_destination | For future use. Must be set to 0. |

Table 3.69. Response

| Byte | Type | Name | Description |
|------|--------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | Unsigned 8-bit integer |

BGScript command

```
call bt_hid_open(address,streaming_destination)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_open(bd_addr *address, uint8 streaming_destination);

/* Response id */
dumo_rsp_bt_hid_open_id

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

Table 3.70. Events Generated

| Event | Description |
|--------------------------------------|---------------------------------------------------------------------------|
| bt_hid_opened | This event indicates the establishment of an HID connection. |
| bt_hid_failed | This event indicates the establishment of an HID connection failed. |
| bt_hid_output_report | This event indicates an HID Output Report was received from the HID Host. |

3.3.1.4 cmd_bt_hid_send_input_report

This command can be used to send HID Input Reports to the HID Host.

Table 3.71. Command

| Byte | Type | Name | Description |
|------|------------|-------------|------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID connection |
| 5 | uint8 | report_id | Report ID, defined in the HID Descriptor |
| 6 | uint8array | report_data | The data payload of the Input Report (prefixed with length byte) |

Table 3.72. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 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 |

BGScript command

```
call bt_hid_send_input_report(endpoint,report_id,report_data_len, report_data_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_send_input_report(uint8 endpoint, uint8 report_id, uint8array report_data);

/* Response id */
dumo_rsp_bt_hid_send_input_report_id

/* Response structure */
struct dumo_msg_bt_hid_send_input_report_rsp_t
{
    uint16 result
}
```

3.3.1.5 cmd_bt_hid_set_report_response

This command is used to acknowledge an evt_bt_hid_set_report request. If the parameters are invalid, use cmd_bt_hid_get_set_report_reject to reject it.

Table 3.73. Command

| Byte | Type | Name | Description |
|------|-------|----------|--------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the request. |

Table 3.74. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 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 |

BGScript command

```
call bt_hid_set_report_response(endpoint)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_set_report_response(uint8 endpoint);

/* Response id */
dumo_rsp_bt_hid_set_report_response_id

/* Response structure */
struct dumo_msg_bt_hid_set_report_response_rsp_t
{
    uint16 result
}
```

3.3.1.6 cmd_bt_hid_start_server

This command can be used to start accepting HID connections.

Table 3.75. Command

| Byte | Type | Name | Description |
|------|-------|-----------------------|------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | sdp_id | ID of the SDP entry defined in project configuration file. |
| 5 | uint8 | streaming_destination | For future use. Must be set to 0. |

Table 3.76. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 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 |

BGScript command

```
call bt_hid_start_server(sdp_id,streaming_destination)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_start_server(uint8 sdp_id, uint8 streaming_destination);

/* Response id */
dumo_rsp_bt_hid_start_server_id

/* Response structure */
struct dumo_msg_bt_hid_start_server_rsp_t
{
    uint16 result
}
```

Table 3.77. Events Generated

| Event | Description |
|---------------|---------------------------------------------------------------------|
| bt_hid_opened | This event indicates the establishment of an HID connection. |
| bt_hid_failed | This event indicates the establishment of an HID connection failed. |

| Event | Description |
|----------------------|---------------------------------------------------------------------------|
| bt_hid_output_report | This event indicates an HID Output Report was received from the HID Host. |

3.3.1.7 cmd_bt_hid_stop_server

This command can be used to stop an HID server.

Table 3.78. Command

| Byte | Type | Name | Description |
|------|-------|--------|----------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | sdp_id | ID of the SDP entry defined in the project configuration file. |

Table 3.79. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 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 |

BGScript command

```
call bt_hid_stop_server(sdp_id)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_stop_server(uint8 sdp_id);

/* Response id */
dumo_rsp_bt_hid_stop_server_id

/* Response structure */
struct dumo_msg_bt_hid_stop_server_rsp_t
{
    uint16 result
}
```

3.3.1.8 cmd_bt_hid_virtual_cable_unplug

This command can be used to disassociate the device from the HID Host. Bonding information will be deleted and the connection disconnected automatically.

Table 3.80. Command

| Byte | Type | Name | Description |
|------|-------|----------|---------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID connection |

Table 3.81. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 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 |

BGScript command

```
call bt_hid_virtual_cable_unplug(endpoint)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_hid_virtual_cable_unplug(uint8 endpoint);

/* Response id */
dumo_rsp_bt_hid_virtual_cable_unplug_id

/* Response structure */
struct dumo_msg_bt_hid_virtual_cable_unplug_rsp_t
{
    uint16 result
}
```

3.3.2 bt_hid events

3.3.2.1 evt_bt_hid_failed

This event indicates the establishment of an HID connection failed.

Table 3.82. Event

| Byte | Type | Name | Description |
|-------|---------|----------|------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x09 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned for the HID connection |
| 5-10 | bd_addr | address | Bluetooth device address of the remote device in little endian format. |
| 11-12 | uint16 | reason | Reason for failure. |

BGScript event

```
event bt_hid_failed(endpoint,address,reason)
```

C Functions

```
/* Event id */
dumo_evt_bt_hid_failed_id

/* Event structure */
struct dumo_msg_bt_hid_failed_evt_t
{
    uint8 endpoint,
    bd_addr *address,
    uint16 reason
}
```

3.3.2.2 evt_bt_hid_get_report

This event indicates the HID Host has requested the current state of a report. This is usually done to determine the initial state of the Device. A response must be sent with cmd_bt_hid_get_report_response. The parameter parameters_used defines which of the parameters were used by the Host to specify the report. The response should be sent with the same parameters_used value.

Table 3.83. Event

| Byte | Type | Name | Description |
|------|--------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the notification. |
| 5 | uint8 | parameters_used | |
| 6 | uint8 | report_type | |
| 7 | uint8 | report_id | ID of the report requested. Must match both report_type and ID, as the same ID can be used for different report types. |
| 8-9 | uint16 | max_bytes | Maximum number of bytes to send, if the report size exceeds the current MTU; the remaining bytes are discarded and not sent in another packet. If a Report ID is sent, it takes up one byte. |

BGScript event

```
event bt_hid_get_report(endpoint,parameters_used,report_type,report_id,max_bytes)
```

C Functions

```
/* Event id */
dumo_evt_bt_hid_get_report_id

/* Event structure */
struct dumo_msg_bt_hid_get_report_evt_t
{
    uint8 endpoint,
    uint8 parameters_used,
    uint8 report_type,
    uint8 report_id,
    uint16 max_bytes
}
```

3.3.2.3 evt_bt_hid_opened

This event indicates the establishment of an HID connection.

Table 3.84. Event

| Byte | Type | Name | Description |
|------|---------|----------|------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x07 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned for the HID connection |
| 5-10 | bd_addr | address | Bluetooth device address of the remote device in little endian format. |

BGScript event

```
event bt_hid_opened(endpoint,address)
```

C Functions

```
/* Event id */
dumo_evt_bt_hid_opened_id

/* Event structure */
struct dumo_msg_bt_hid_opened_evt_t
{
    uint8 endpoint,
    bd_addr *address
}
```

3.3.2.4 evt_bt_hid_output_report

This event indicates an HID Output Report was received from the HID Host.

Table 3.85. Event

| Byte | Type | Name | Description |
|------|------------|-------------|-------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the report. |
| 5 | uint8 | report_id | Report ID defined in the HID Descriptor. |
| 6 | uint8array | report_data | Payload of the HID Output Report (prefixed by length byte). |

BGScript event

```
event bt_hid_output_report(endpoint,report_id,report_data_len, report_data_data)
```

C Functions

```
/* Event id */
dumo_evt_bt_hid_output_report_id

/* Event structure */
struct dumo_msg_bt_hid_output_report_evt_t
{
    uint8 endpoint,
    uint8 report_id,
    uint8array report_data
}
```

3.3.2.5 evt_bt_hid_set_report

This event indicates an HID report was received from the HID Host.

Table 3.86. Event

| Byte | Type | Name | Description |
|------|------------|-------------|-------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the report. |
| 5 | uint8 | report_type | |
| 6 | uint8 | report_id | Report ID defined in the HID Descriptor. |
| 7 | uint8array | report_data | Payload of the HID Report (prefixed by length byte). |

BGScript event

```
event bt_hid_set_report(endpoint,report_type,report_id,report_data_len, report_data_data)
```

C Functions

```
/* Event id */
dumo_evt_bt_hid_set_report_id

/* Event structure */
struct dumo_msg_bt_hid_set_report_evt_t
{
    uint8 endpoint,
    uint8 report_type,
    uint8 report_id,
    uint8array report_data
}
```

3.3.2.6 evt_bt_hid_state_changed

This event indicates the HID Host has notified the Device about a state change. No response is required.

Table 3.87. Event

| Byte | Type | Name | Description |
|------|-------|----------|-------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x13 | class | Message class: HID |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle of the HID Host that sent the notification. |
| 5 | uint8 | state | |

BGScript event

```
event bt_hid_state_changed(endpoint,state)
```

C Functions

```
/* Event id */
dumo_evt_bt_hid_state_changed_id

/* Event structure */
struct dumo_msg_bt_hid_state_changed_evt_t
{
    uint8 endpoint,
    uint8 state
}
```

3.3.3 bt_hid enumerations

3.3.3.1 enum_bt_hid_get_report_params

These values indicate which parameters should be ignored in the bt_hid_get_report events. The response should be sent with the same value.

Table 3.88. Enumerations

| Value | Name | Description |
|-------|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | bt_hid_params_no_id_no_max_size | The Host has requested the current state for the report unambiguously specified by Report Type, with no maximum data size constraints. Report ID and Max Bytes should be ignored. |
| 1 | bt_hid_params_with_id_no_max_size | The Host has requested the current state for the report identified by Report Type and Report ID, with no maximum data size constraints. Max Bytes should be ignored. |
| 2 | bt_hid_params_no_id_with_max_size | The Host has requested the current state for the report unambiguously specified by Report Type. Report ID should be ignored. Only the first N bytes of the report, as specified by Max Bytes, shall be sent. |
| 3 | bt_hid_params_with_id_with_max_size | The Host has requested the current state for the report identified by Report Type and Report ID. Only the first N-1 bytes of the report, as specified by Max Bytes, shall be sent. (Report ID will take up 1 byte.) |

3.3.3.2 enum_bt_hid_report_type

Table 3.89. Enumerations

| Value | Name | Description |
|-------|----------------------------|----------------|
| 0 | bt_hid_report_type_other | Other Report |
| 1 | bt_hid_report_type_input | Input Report |
| 2 | bt_hid_report_type_output | Output Report |
| 3 | bt_hid_report_type_feature | Feature Report |

3.3.3.3 enum_bt_hid_state

These values describe state changes for HID connections.

Table 3.90. Enumerations

| Value | Name | Description |
|-------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | bt_hid_state_boot_mode | The Host has set the Device to use only Boot Reports. The send_input_report command should not be used in this state. |
| 1 | bt_hid_state_report_mode | The Host has set the Device to use Report Mode. The send_boot_report command should not be used in this state. This is the default state a HID connection is in. |
| 3 | bt_hid_state_suspend | The Host has entered a state where normal operation is no longer required from the Device, for example gone to sleep mode. The HID Device may also enter a power saving mode or even disconnect from the Host. |
| 4 | bt_hid_state_exit_suspend | The Host has resumed normal operation and the Device should exit any power saving mode it may have been in. |
| 5 | bt_hid_state_virtual_cable_unplug | The Host has requested to disassociate the Device with the Host, and will delete the bonding information it has associated with the Device. The connection will be automatically closed by the HID Device and its corresponding bonding deleted. |

3.4 RFCOMM (bt_rfcomm)

These commands and events are related to the establishment of Bluetooth Classic RFCOMM connections as specified in the Serial Port Profile SPP).

3.4.1 bt_rfcomm commands

3.4.1.1 cmd_bt_rfcomm_modem_status

This command sets modem control status bits for RFCOMM connection. This is a legacy support feature and **SHOULD NOT BE USED FOR FLOW CONTROL**. For description and meaning of bits in this field see the ETSI TS 07.10 specification.

Table 3.91. Command

| Byte | Type | Name | Description |
|------|-------|----------|-----------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned to the RFCOMM connection |
| 5 | uint8 | modem | New values of the modem status bits. |
| 6 | uint8 | mask | Bitmask of which bits are changed. |

Table 3.92. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 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 |

BGScript command

```
call bt_rfcomm_modem_status(endpoint,modem,mask)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_modem_status(uint8 endpoint, uint8 modem, uint8 mask);

/* Response id */
dumo_rsp_bt_rfcomm_modem_status_id

/* Response structure */
struct dumo_msg_bt_rfcomm_modem_status_rsp_t
{
    uint16 result
}
```

3.4.1.2 cmd_bt_rfcomm_open

Open an RFCOMM channel to remote device. An existing connection, or a connection attempt before the page timeout expires, can be closed with the [endpoint_close](#) command. NOTE: Controller does not allow starting a Classic BR/EDR connection while a classic discovery process is in progress. Please terminate any classic discovery with [bt_gap_cancel_discovery](#) command prior to starting the RFCOMM connection.

Table 3.93. Command

| Byte | Type | Name | Description |
|------|------------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of the remote device in little endian format |
| 10 | uint8 | streaming_destination | Streaming destination for the connection endpoint. Endpoint 'SCRIPT' is a special case in the BGAPI mode for RFCOMM. Data events are sent to script and in addition also to UART. Please note that if size event exceeds size of 256 (inclusive of headers) it will be rejected by the script interpreter. |
| 11 | uint8array | uuid | UUID of the profile to connect to: <ul style="list-style-type: none"> • uuid16, 2 bytes: 16-bit UUID for searching RFCOMM port • uuid32, 4 bytes: 32-bit UUID for searching RFCOMM port • uuid128, 16 bytes: 128-bit UUID for searching RFCOMM port |

Table 3.94. Response

| Byte | Type | Name | Description |
|------|--------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | Unsigned 8-bit integer |

BGScript command

```
call bt_rfcomm_open(address,streaming_destination,uuid_len, uuid_data)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_open(bd_addr *address, uint8 streaming_destination, uint8array uuid);

/* Response id */
dumo_rsp_bt_rfcomm_open_id

/* Response structure */
```

```
struct dumo_msg_bt_rfcomm_open_rsp_t
{
    uint16 result,
    uint8 endpoint
}
```

Table 3.95. Events Generated

| Event | Description |
|-------------------------------|-----------------------------------------------------------------|
| <code>bt_rfcomm_opened</code> | This event indicates the establishment of an RFCOMM connection. |

3.4.1.3 cmd_bt_rfcomm_open_port

This command can be used to open an RFCOMM channel to a remote device using a fixed RFCOMM port number. This bypasses the Service Discovery Protocol procedure for retrieving the RFCOMM port from the remote device's Serial Port Profile SDP entry, thus speeding up connection establishment.

Table 3.96. Command

| Byte | Type | Name | Description |
|------|---------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x02 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of the remote device in little endian format |
| 10 | uint8 | streaming_destination | Streaming destination for the connection endpoint. Endpoint 'SCRIPT' is a special case in the BGAPI mode for RFCOMM. Data events are sent to script and in addition also to UART. Please note that if size event exceeds size of 256 (inclusive of headers) it will be rejected by the script interpreter. |
| 11 | uint8 | port | RFCOMM port |

Table 3.97. Response

| Byte | Type | Name | Description |
|------|--------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | Unsigned 8-bit integer |

BGScript command

```
call bt_rfcomm_open_port(address,streaming_destination,port)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_open_port(bd_addr *address, uint8 streaming_destination, uint8 port);

/* Response id */
dumo_rsp_bt_rfcomm_open_port_id

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

Table 3.98. Events Generated

| Event | Description |
|------------------|-----------------------------------------------------------------|
| bt_rfcomm_opened | This event indicates the establishment of an RFCOMM connection. |

3.4.1.4 cmd_bt_rfcomm_set_modem_status_default

This command sets starting value modem control status bits for new outgoing RFCOMM connections.

Table 3.99. Command

| Byte | Type | Name | Description |
|------|-------|--------|-------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | modem | Starting values of the modem status bits. |

Table 3.100. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 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 |

BGScript command

```
call bt_rfcomm_set_modem_status_default(modem)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_set_modem_status_default(uint8 modem);

/* Response id */
dumo_rsp_bt_rfcomm_set_modem_status_default_id

/* Response structure */
struct dumo_msg_bt_rfcomm_set_modem_status_default_rsp_t
{
    uint16 result
}
```

3.4.1.5 cmd_bt_rfcomm_start_server

This command can be used to start a RFCOMM server as defined in the referenced SDP-entry.

Table 3.101. Command

| Byte | Type | Name | Description |
|------|-------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | sdp_id | ID of the SDP entry defined in project configuration file |
| 5 | uint8 | streaming_destination | Streaming destination endpoint for the connection. This should be one of the fixed endpoints. While a dynamic connection endpoint will work for proxying data coming from an RFCOMM connection to another connection endpoint, there is no guarantee that the destination endpoint will be there when an incoming RFCOMM connection is established. Endpoint 'SCRIPT' is a special case in the BGAPI mode for RFCOMM. Data events are sent to script and in addition also to UART. Please note that if size event exceeds size of 256 (inclusive of headers) it will be rejected by the script interpreter. |

Table 3.102. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_rfcomm_start_server(sdp_id,streaming_destination)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_start_server(uint8 sdp_id, uint8 streaming_destination);

/* Response id */
dumo_rsp_bt_rfcomm_start_server_id

/* Response structure */
struct dumo_msg_bt_rfcomm_start_server_rsp_t
{
    uint16 result
}
```

3.4.1.6 cmd_bt_rfcomm_start_server_port

This command can be used to start a RFCOMM server on specific port. In contrast to [cmd_bt_rfcomm_start_server](#) command, components of the port descriptor are not loaded from SDP records.

Table 3.103. Command

| Byte | Type | Name | Description |
|------|-------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | port | port to use |
| 5 | uint8 | streaming_destination | Streaming destination endpoint for the connection. This should be one of the fixed endpoints. While a dynamic connection endpoint will work for proxying data coming from an RFCOMM connection to another connection endpoint, there is no guarantee that the destination endpoint will be there when an incoming RFCOMM connection is established. |

Table 3.104. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 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 |

BGScript command

```
call bt_rfcomm_start_server_port(port,streaming_destination)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_start_server_port(uint8 port, uint8 streaming_destination);

/* Response id */
dumo_rsp_bt_rfcomm_start_server_port_id

/* Response structure */
struct dumo_msg_bt_rfcomm_start_server_port_rsp_t
{
    uint16 result
}
```

3.4.1.7 cmd_bt_rfcomm_stop_server

This command can be used to stop an RFCOMM server.

Table 3.105. Command

| Byte | Type | Name | Description |
|------|-------|--------|----------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | sdp_id | ID of the SDP entry defined in the project configuration file. |

Table 3.106. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 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 |

BGScript command

```
call bt_rfcomm_stop_server(sdp_id)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_stop_server(uint8 sdp_id);

/* Response id */
dumo_rsp_bt_rfcomm_stop_server_id

/* Response structure */
struct dumo_msg_bt_rfcomm_stop_server_rsp_t
{
    uint16 result
}
```

3.4.1.8 cmd_bt_rfcomm_stop_server_port

This command can be used to stop an RFCOMM server that has been started on a port.

Table 3.107. Command

| Byte | Type | Name | Description |
|------|-------|--------|---------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | port | port server is listening. |

Table 3.108. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 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 |

BGScript command

```
call bt_rfcomm_stop_server_port(port)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_rfcomm_stop_server_port(uint8 port);

/* Response id */
dumo_rsp_bt_rfcomm_stop_server_port_id

/* Response structure */
struct dumo_msg_bt_rfcomm_stop_server_port_rsp_t
{
    uint16 result
}
```

3.4.2 bt_rfcomm events

3.4.2.1 evt_bt_rfcomm_credit_starvation

This event indicates that outgoing data traffic has possibly been stuck. If there is no valid reason why the other end has not sent us flow control credits then the connection should be closed.

Table 3.109. Event

| Byte | Type | Name | Description |
|------|-------|----------|-----------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned to the RFCOMM connection |

BGScript event

```
event bt_rfcomm_credit_starvation(endpoint)
```

C Functions

```
/* Event id */
dumo_evt_bt_rfcomm_credit_starvation_id

/* Event structure */
struct dumo_msg_bt_rfcomm_credit_starvation_evt_t
{
    uint8 endpoint
}
```

3.4.2.2 evt_bt_rfcomm_incoming_rejected

This event indicates that incoming rfcomm connection was rejected due being low in memory.

Table 3.110. Event

| Byte | Type | Name | Description |
|------|---------|---------|----------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x02 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of the remote device in little endian format |

BGScript event

```
event bt_rfcomm_incoming_rejected(address)
```

C Functions

```
/* Event id */
dumo_evt_bt_rfcomm_incoming_rejected_id

/* Event structure */
struct dumo_msg_bt_rfcomm_incoming_rejected_evt_t
{
    bd_addr *address
}
```

3.4.2.3 evt_bt_rfcomm_modem_status

This event indicates change in modem control signals.

Table 3.111. Event

| Byte | Type | Name | Description |
|------|-------|----------|-------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned to the RFCOMM connection |
| 5 | uint8 | modem | Value of the modem status. For description and meaning of bits in this field see ETSI TS 07.10. |

BGScript event

```
event bt_rfcomm_modem_status(endpoint,modem)
```

C Functions

```
/* Event id */
dumo_evt_bt_rfcomm_modem_status_id

/* Event structure */
struct dumo_msg_bt_rfcomm_modem_status_evt_t
{
    uint8 endpoint,
    uint8 modem
}
```

3.4.2.4 evt_bt_rfcomm_opened

This event indicates the establishment of an RFCOMM connection.

Table 3.112. Event

| Byte | Type | Name | Description |
|------|---------|----------|-----------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x07 | olen | Minimum payload length |
| 2 | 0x04 | class | Message class: RFCOMM |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned to the RFCOMM connection |
| 5-10 | bd_addr | address | Bluetooth address of the remote Bluetooth Classic device in little endian format. |

BGScript event

```
event bt_rfcomm_opened(endpoint,address)
```

C Functions

```
/* Event id */
dumo_evt_bt_rfcomm_opened_id

/* Event structure */
struct dumo_msg_bt_rfcomm_opened_evt_t
{
    uint8 endpoint,
    bd_addr *address
}
```

3.5 Service Discovery Profile (bt_sdp)

The commands and events in this are related to the Service Discovery Profile (SDP).

3.5.1 bt_sdp commands

3.5.1.1 cmd_bt_sdp_add_entry

This function load custom defined SDP record into the SDP server. Note that this funtion does not verify the correctness of the record at all.

Table 3.113. Command

| Byte | Type | Name | Description |
|------|------------|--------|-----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | lolen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x02 | method | Message ID |
| 4-7 | uint32 | sdp_id | ID of the SDP entry to be loaded. Range from 1000 to 2000 |
| 8 | uint8array | record | SDP record |

Table 3.114. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery 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 |

BGScript command

```
call bt_sdp_add_entry(sdp_id,record_len, record_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_sdp_add_entry(uint32 sdp_id, uint8array record);

/* Response id */
dumo_rsp_bt_sdp_add_entry_id

/* Response structure */
struct dumo_msg_bt_sdp_add_entry_rsp_t
{
    uint16 result
}
```

3.5.1.2 cmd_bt_sdp_delete_entry

This function removes custom defined SDP record from the SDP server.

Table 3.115. Command

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x03 | method | Message ID |
| 4-7 | uint32 | sdp_id | ID of the SDP entry to be loaded. Range from 1000 to 2000 |

Table 3.116. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery 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 |

BGScript command

```
call bt_sdp_delete_entry(sdp_id)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_sdp_delete_entry(uint32 sdp_id);

/* Response id */
dumo_rsp_bt_sdp_delete_entry_id

/* Response structure */
struct dumo_msg_bt_sdp_delete_entry_rsp_t
{
    uint16 result
}
```

3.5.1.3 cmd_bt_sdp_load_entry

This command can be used to load an SDP entry from the internal filesystem to the SDP server. NOTE: command only loads SDP record. It does not do anything else. Like start RFCOMM server

Table 3.117. Command

| Byte | Type | Name | Description |
|------|--------|--------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x04 | olen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x01 | method | Message ID |
| 4-7 | uint32 | sdp_id | ID of the SDP entry defined in the project configuration file. See the Bluetooth Smart Ready Serial Port Profile Application Notes for details |

Table 3.118. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_sdp_load_entry(sdp_id)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_sdp_load_entry(uint32 sdp_id);

/* Response id */
dumo_rsp_bt_sdp_load_entry_id

/* Response structure */
struct dumo_msg_bt_sdp_load_entry_rsp_t
{
    uint16 result
}
```

3.5.1.4 cmd_bt_sdp_search_rfcomm_port

This command can be used to search for Serial Port Profile (SPP) services in a Bluetooth Classic device. The reply is an event specifying the RFCOMM port found.

Table 3.119. Command

| Byte | Type | Name | Description |
|------|------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth device address of the remote device in little endian format |
| 10 | uint8array | uuid | UUID is one of the following: <ul style="list-style-type: none"> • uuid16, 2 bytes: 16-bit UUID for searching RFCOMM port • uuid32, 4 bytes: 32-bit UUID for searching RFCOMM port • uuid128, 16 bytes: 128-bit UUID for RFCOMM port |

Table 3.120. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call bt_sdp_search_rfcomm_port(address,uuid_len, uuid_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_bt_sdp_search_rfcomm_port(bd_addr *address, uint8array uuid);

/* Response id */
dumo_rsp_bt_sdp_search_rfcomm_port_id

/* Response structure */
struct dumo_msg_bt_sdp_search_rfcomm_port_rsp_t
{
    uint16 result
}
```

Table 3.121. Events Generated

| Event | Description |
|-------------------------------------------|---------------------------------------------|
| bt_sdp_search_rfcomm_port | This event indicates the SDP search result. |

3.5.2 bt_sdp events**3.5.2.1 evt_bt_sdp_search_rfcomm_port**

This event indicates the SDP search result.

Table 3.122. Event

| Byte | Type | Name | Description |
|------|------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x08 | olen | Minimum payload length |
| 2 | 0x06 | class | Message class: Service Discovery Profile |
| 3 | 0x00 | method | Message ID |
| 4-7 | uint32 | request_handle | Reserved; should be set to zero |
| 8-9 | uint16 | status | Result code • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 10 | uint8 | port | RFCOMM port number |
| 11 | uint8array | service_name | RFCOMM service name |

BGScript event

```
event bt_sdp_search_rfcomm_port(request_handle,status,port,service_name_len, service_name_data)
```

C Functions

```
/* Event id */
dumo_evt_bt_sdp_search_rfcomm_port_id

/* Event structure */
struct dumo_msg_bt_sdp_search_rfcomm_port_evt_t
{
    uint32 request_handle,
    uint16 status,
    uint8 port,
    uint8array service_name
}
```

3.6 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 module 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 [Flash Upload Finish](#) when all the data has been uploaded
6. Finalize the DFU firmware update with [Reset command](#).

Bootloader checks the CRC checksum from the uploaded image and the module will remain in DFU mode if it does not match. DFU mode is using UART baudrate from the hardware configuration of the firmware. Default baudrate 115200 is used if firmware is missing or firmware content does not match with CRC checksum.

3.6.1 dfu commands

3.6.1.1 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 3.123. Command

| Byte | Type | Name | Description |
|------|--------|---------|-------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | 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 3.124. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | 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 |

BGScript command

```
call dfu_flash_set_address(address)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_dfu_flash_set_address(uint32 address);

/* Response id */
dumo_rsp_dfu_flash_set_address_id

/* Response structure */
struct dumo_msg_dfu_flash_set_address_rsp_t
{
    uint16 result
}
```

3.6.1.2 cmd_dfu_flash_upload

This command can be used to upload the whole firmware image file in to the Bluetooth module. The recommended payload size of the command is 128 bytes, so multiple commands need to be issued one after the other until the whole .bin firmware image file is uploaded to the module. The next address of the flash sector in memory to write to is automatically updated by the bootloader after each individual command.

Table 3.125. Command

| Byte | Type | Name | Description |
|------|------------|--------|--------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | olen | 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 3.126. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call dfu_flash_upload(data_len, data_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_dfu_flash_upload(uint8array data);

/* Response id */
dumo_rsp_dfu_flash_upload_id

/* Response structure */
struct dumo_msg_dfu_flash_upload_rsp_t
{
    uint16 result
}
```

3.6.1.3 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 3.127. Command

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

Table 3.128. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | 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 |

BGScript command

```
call dfu_flash_upload_finish()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_dfu_flash_upload_finish();

/* Response id */
dumo_rsp_dfu_flash_upload_finish_id

/* Response structure */
struct dumo_msg_dfu_flash_upload_finish_rsp_t
{
    uint16 result
}
```

3.6.1.4 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 3.129. Command

| Byte | Type | Name | Description |
|------|-------|--------|----------------------------------------------------------|
| 0 | 0x20 | hlen | 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 DFU mode |

BGScript command

```
call dfu_reset(dfu)
```

BGLIB C API

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

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

Table 3.130. Events Generated

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

3.6.2 dfu events

3.6.2.1 evt_dfu_boot

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

Table 3.131. Event

| Byte | Type | Name | Description |
|------|--------|---------|----------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x04 | olen | 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 |

BGScript event

```
event dfu_boot(version)
```

C Functions

```
/* Event id */
dumo_evt_dfu_boot_id

/* Event structure */
struct dumo_msg_dfu_boot_evt_t
{
    uint32 version
}
```

3.7 Endpoint (endpoint)

These commands and events are related to the control of endpoints. They allow the creation and deletion of endpoints as well as configuration of data routing. Predefined endpoints ID's are:

- **0: UART**
- **1: SCRIPT**
- **3: SPI1** (only if defined in hardware configuration)
- **4: SPI2** (only if defined in hardware configuration)
- **31: DROP**

Note the difference between these endpoint ID's and the [endpoint types](#) is that endpoint types describe different categories. Each endpoint has an endpoint type which describes what kind of endpoint it is. There may be multiple endpoints which have the same type. Each endpoint has exactly one ID, and each ID points to exactly one endpoint.

3.7.1 endpoint commands

3.7.1.1 cmd_endpoint_close

This command can be used to close an RFCOMM endpoint , LE cable replacement endpoint, or a BLE connection. This command must always be used to close an endpoint with WAIT_CLOSE status when the remote side has closed the RFCOMM or LE cable replacement connection. This is to free the memory allocated by the endpoint for future reuse in the case when the remote side closes the connection. Note : this command may not be issued inside `endpoint_data` event handler in BgScript.

Table 3.132. Command

| Byte | Type | Name | Description |
|------|-------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | endpoint | The index of the RFCOMM/LE cable replacement endpoint to close or the connection handle ID of the BLE connection to be closed. The connection handle ID is reported for example in the event <code>le_connection_opened</code> . |

Table 3.133. Response

| Byte | Type | Name | Description |
|------|--------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | The endpoint that was closed |

BGScript command

```
call endpoint_close(endpoint)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_endpoint_close(uint8 endpoint);

/* Response id */
dumo_rsp_endpoint_close_id

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

Table 3.134. Events Generated

| Event | Description |
|---------------------------------|-----------------------------------|
| endpoint_status | Sent when endpoint status changes |

3.7.1.2 cmd_endpoint_read_counters

This command can be used to read the data performance counters (data sent counter and data received counter) of an endpoint.

Table 3.135. Command

| Byte | Type | Name | Description |
|------|-------|----------|-------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | endpoint | The endpoint's handle |

Table 3.136. Response

| Byte | Type | Name | Description |
|-------|--------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 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 | uint8 | endpoint | |
| 7-10 | uint32 | tx | Amount of data sent to this endpoint |
| 11-14 | uint32 | rx | Amount of data received from this endpoint |

BGScript command

```
call endpoint_read_counters(endpoint)(result,endpoint,tx,rx)
```

BGLIB C API

```
/* Function */
void dumo_cmd_endpoint_read_counters(uint8 endpoint);

/* Response id */
dumo_rsp_endpoint_read_counters_id

/* Response structure */
struct dumo_msg_endpoint_read_counters_rsp_t
{
    uint16 result,
    uint8 endpoint,
    uint32 tx,
    uint32 rx
}
```

3.7.1.3 cmd_endpoint_send

This command can be used to send data to the defined endpoint. When this command is issued data to be sent is placed to send queue for sending, except for SPI endpoints. In case of SPI endpoints data is immediately sent over the interface and transfer is complete once the response to this command has been received.

Table 3.137. Command

| Byte | Type | Name | Description |
|------|------------|----------|-----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | endpoint | The index of the endpoint to which the data will be sent. |
| 5 | uint8array | data | The RAW data which will be written or sent |

Table 3.138. Response

| Byte | Type | Name | Description |
|------|--------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code • 0: success • 0x018e: Internal buffers are full and the data was discarded and not sent. The host should resend the data again later. The host can optionally wait for an endpoint_status event and check that endpoint_FLAG_FULL is not set. • Non-zero: an other error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | The endpoint to which the data was written |

BGScript command

```
call endpoint_send(endpoint,data_len, data_data)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_endpoint_send(uint8 endpoint, uint8array data);

/* Response id */
dumo_rsp_endpoint_send_id

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

3.7.1.4 cmd_endpoint_set_active

This command can be used to set the endpoint active or inactive. When the endpoint is inactive, any incoming data will be paused, and no data may be sent out. Not all endpoint types can be set inactive. Currently only rfcomm endpoints can be set inactive.

Table 3.139. Command

| Byte | Type | Name | Description |
|------|-------|----------|-------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | endpoint | The endpoint's handle |
| 5 | uint8 | active | Active state (0 or 1). |

Table 3.140. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 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 |

BGScript command

```
call endpoint_set_active(endpoint,active)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_endpoint_set_active(uint8 endpoint, uint8 active);

/* Response id */
dumo_rsp_endpoint_set_active_id

/* Response structure */
struct dumo_msg_endpoint_set_active_rsp_t
{
    uint16 result
}
```

Table 3.141. Events Generated

| Event | Description |
|-----------------|--------------------------------------------|
| endpoint_status | This event indicates an endpoint's status. |

3.7.1.5 cmd_endpoint_set_streaming_destination

This command can be used to set the destination into which data from an endpoint will be routed to.

Table 3.142. Command

| Byte | Type | Name | Description |
|------|-------|----------------------|-------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | endpoint | The endpoint which to control |
| 5 | uint8 | destination_endpoint | The destination for the data |

Table 3.143. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 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 |

BGScript command

```
call endpoint_set_streaming_destination(endpoint,destination_endpoint)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_endpoint_set_streaming_destination(uint8 endpoint, uint8 destination_endpoint);

/* Response id */
dumo_rsp_endpoint_set_streaming_destination_id

/* Response structure */
struct dumo_msg_endpoint_set_streaming_destination_rsp_t
{
    uint16 result
}
```

Table 3.144. Events Generated

| Event | Description |
|---------------------------------|-----------------------------------|
| endpoint_status | Sent when endpoint status changes |

3.7.2 endpoint events

3.7.2.1 evt_endpoint_closed

This event indicates that an RFCOMM endpoint has been closed by the safety timer. After endpoint_closing event user should close the endpoint to release the resources taken by the endpoint. If user fails to do so in about 5 seconds then module SW will release resources and dispatch this event.

Table 3.145. Event

| Byte | Type | Name | Description |
|------|-------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint handle. Values: <ul style="list-style-type: none">• 0xff: connection failed without associated endpoint• Others: Handle for closed endpoint |

BGScript event

```
event endpoint_closed(endpoint)
```

C Functions

```
/* Event id */
dumo_evt_endpoint_closed_id

/* Event structure */
struct dumo_msg_endpoint_closed_evt_t
{
    uint8 endpoint
}
```

3.7.2.2 evt_endpoint_closing

This event indicates that an endpoint is closing or that the remote end has terminated the connection. This event should be acknowledged by calling the [endpoint_close](#) command or otherwise the firmware will not re-use the endpoint index.

Table 3.146. Event

| Byte | Type | Name | Description |
|------|--------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0xb0 | class | Message class: Endpoint |
| 3 | 0x03 | 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 | endpoint | Endpoint handle. Values: • 0xff: connection failed without associated endpoint • Others: Handle for closed endpoint |

BGScript event

```
event endpoint_closing(reason,endpoint)
```

C Functions

```
/* Event id */
dumo_evt_endpoint_closing_id

/* Event structure */
struct dumo_msg_endpoint_closing_evt_t
{
    uint16 reason,
    uint8 endpoint
}
```

3.7.2.3 evt_endpoint_data

This event indicates incoming data from an endpoint. Note : `endpoint_close` command must not be used in BgScript handler for this event.

Table 3.147. Event

| Byte | Type | Name | Description |
|------|------------|----------|-------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0b | class | Message class: Endpoint |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | endpoint | The endpoint which received this data, i.e. to which it was sent. |
| 5 | uint8array | data | The raw data |

BGScript event

```
event endpoint_data(endpoint,data_len, data_data)
```

C Functions

```
/* Event id */
dumo_evt_endpoint_data_id

/* Event structure */
struct dumo_msg_endpoint_data_evt_t
{
    uint8 endpoint,
    uint8array data
}
```

3.7.2.4 evt_endpoint_status

This event indicates an endpoint's status.

Table 3.148. Event

| Byte | Type | Name | Description |
|------|--------|----------------------|-------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x07 | olen | Minimum payload length |
| 2 | 0xb0 | class | Message class: Endpoint |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | endpoint | The index of the endpoint whose status this event describes |
| 5-8 | uint32 | type | Unsigned 32-bit integer |
| 9 | int8 | destination_endpoint | The index of the endpoint to which the incoming data goes. |
| 10 | uint8 | flags | Flags which indicate the mode of endpoint |

BGScript event

```
event endpoint_status(endpoint,type,destination_endpoint,flags)
```

C Functions

```
/* Event id */
dumo_evt_endpoint_status_id

/* Event structure */
struct dumo_msg_endpoint_status_evt_t
{
    uint8 endpoint,
    uint32 type,
    int8 destination_endpoint,
    uint8 flags
}
```

3.7.2.5 evt_endpoint_syntax_error

This event indicates that a protocol error was detected in BGAPI command parser. This event is triggered if a BGAPI command from the host contains syntax error(s), or if a command is only partially sent, in which case the BGAPI parser has a 1 second command timeout by default. In practice, the latter case happens if a byte belonging to a command is sent not before a timeout of 1 second since the previous byte. The default timeout can be changed from 1ms to 4s via the "timeout" parameter of the attribute <uart> in the hardware.xml firmware project file. Any partial or wrongly formatted command will be discarded.

Table 3.149. Event

| Byte | Type | Name | Description |
|------|--------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0xb0 | class | Message class: Endpoint |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | Endpoint of the connection |

BGScript event

```
event endpoint_syntax_error(result,endpoint)
```

C Functions

```
/* Event id */
dumo_evt_endpoint_syntax_error_id

/* Event structure */
struct dumo_msg_endpoint_syntax_error_evt_t
{
    uint16 result,
    uint8 endpoint
}
```

3.7.3 endpoint enumerations

3.7.3.1 enum_endpoint_types

These values define the endpoint types. Please note that there may be multiple endpoints with type

Table 3.150. Enumerations

| Value | Name | Description |
|-------|---------------------|-------------------------------------|
| 0 | endpoint_free | Endpoint is not in use |
| 1 | endpoint_uart | UART |
| 2 | endpoint_script | Scripting |
| 4 | endpoint_reserved | Reserved for future use |
| 16 | endpoint_drop | Drop all data sent to this endpoint |
| 32 | endpoint_rfcomm | RFCOMM channel |
| 64 | endpoint_spi | SPI |
| 128 | endpoint_connection | Connection |
| 512 | endpoint_iap | iAP |
| 1024 | endpoint_l2cap | L2CAP |
| 2048 | endpoint_hid | HID |
| 4096 | endpoint_leserial | Bluetooth LE serial channel |

3.7.4 endpoint defines

3.7.4.1 define_endpoint_endpoint_flags

Table 3.151. Defines

| Value | Name | Description |
|-------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | ENDPOINT_FLAG_UPDATED | Endpoint status has been changed since last indication |
| 2 | ENDPOINT_FLAG_ACTIVE | Endpoint is active and can send and receive data |
| 4 | ENDPOINT_FLAG_STREAMING | Endpoint is in streaming mode. Data is sent and received from endpoint without framing |
| 8 | ENDPOINT_FLAG_BGAPI | Endpoint is configured for BGAPI. Data received is parsed as BGAPI commands. Also all BGAPI events and responses are sent to this endpoint |
| 16 | ENDPOINT_FLAG_WAIT_CLOSE | Endpoint is closed from the device side. Host needs to acknowledge by sending endpoint_close command to device with this endpoint as parameter |
| 32 | ENDPOINT_FLAG_CLOSING | Endpoint is closed from the host side and is waiting for the device to acknowledge closing of the endpoint |
| 64 | ENDPOINT_FLAG_FULL | Endpoint buffers are full and no data from host can be sent to it. When more data can be sent to the endpoint, a new endpoint_status event will be generated with the updated flags. |

3.8 Persistent Store (flash)

These commands and events can be used to manage the user data in the flash memory of the Bluetooth module. User data stored in PS keys within the flash memory is persistent across reset and power cycling of the module.

3.8.1 flash commands

3.8.1.1 cmd_flash_ps_dump

This command can be used to retrieve all PS keys and their current values. For each existing PS key a flash_pskey event will be generated which includes the corresponding PS key value.

Table 3.152. Command

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0xd0 | class | Message class: Persistent Store |
| 3 | 0x00 | method | Message ID |

Table 3.153. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0xd0 | class | Message class: Persistent Store |
| 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 |

BGScript command

```
call flash_ps_dump()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_flash_ps_dump();

/* Response id */
dumo_rsp_flash_ps_dump_id

/* Response structure */
struct dumo_msg_flash_ps_dump_rsp_t
{
    uint16 result
}
```

Table 3.154. Events Generated

| Event | Description |
|------------------------------|-----------------|
| flash_ps_key | PS Key contents |

3.8.1.2 cmd_flash_ps_erase

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

Table 3.155. Command

| Byte | Type | Name | Description |
|------|--------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0xd0 | class | Message class: Persistent Store |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | key | PS key to erase |

Table 3.156. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0xd0 | 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 |

BGScript command

```
call flash_ps_erase(key)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_flash_ps_erase(uint16 key);

/* Response id */
dumo_rsp_flash_ps_erase_id

/* Response structure */
struct dumo_msg_flash_ps_erase_rsp_t
{
    uint16 result
}
```

3.8.1.3 cmd_flash_ps_erase_all

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

Table 3.157. Command

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0xd0 | class | Message class: Persistent Store |
| 3 | 0x01 | method | Message ID |

Table 3.158. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0xd0 | 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 |

BGScript command

```
call flash_ps_erase_all()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_flash_ps_erase_all();

/* Response id */
dumo_rsp_flash_ps_erase_all_id

/* Response structure */
struct dumo_msg_flash_ps_erase_all_rsp_t
{
    uint16 result
}
```

3.8.1.4 cmd_flash_ps_load

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

Table 3.159. Command

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0xd0 | class | Message class: Persistent Store |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | key | PS key of the value to be retrieved |

Table 3.160. Response

| Byte | Type | Name | Description |
|------|------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0xd0 | 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. |

BGScript command

```
call flash_ps_load(key)(result,value_len, value_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_flash_ps_load(uint16 key);

/* Response id */
dumo_rsp_flash_ps_load_id

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

3.8.1.5 cmd_flash_ps_save

This command can be used to store a value into the specified PS key. The allowed PS key range is from 0x4000 to 0x407F. Maximum length of value is 250 bytes. Maximum storage space is 4 KB and this is shared with system internal keys. Available storage space depends on how much storage space the application will use. In addition 20 bytes of metadata is stored for each PS item.

Table 3.161. 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 3.162. 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 |

BGScript command

```
call flash_ps_save(key,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_flash_ps_save(uint16 key, uint8array value);

/* Response id */
dumo_rsp_flash_ps_save_id

/* Response structure */
struct dumo_msg_flash_ps_save_rsp_t
{
    uint16 result
}
```

3.8.2 flash events

3.8.2.1 evt_flash_ps_key

This event indicates that the flash_ps_dump command was given. It returns a single PS key and its value. There can be multiple events if multiple PS keys exist.

Table 3.163. Event

| Byte | Type | Name | Description |
|------|------------|--------|------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0xd0 | class | Message class: Persistent Store |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | key | PS key |
| 6 | uint8array | value | Current value of the PS key specified in this event. |

BGScript event

```
event flash_ps_key(key,value_len, value_data)
```

C Functions

```
/* Event id */
dumo_evt_flash_ps_key_id

/* Event structure */
struct dumo_msg_flash_ps_key_evt_t
{
    uint16 key,
    uint8array value
}
```

3.9 Generic Attribute Profile (gatt)

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

3.9.1 gatt commands

3.9.1.1 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 successfully completed or failed with error.

Table 3.164. Command

| Byte | Type | Name | Description |
|------|--------|------------|---------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x05 | olen | 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 3.165. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_discover_characteristics(connection,service)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_discover_characteristics(uint8 connection, uint32 service);

/* Response id */
dumo_rsp_gatt_discover_characteristics_id

/* Response structure */
struct dumo_msg_gatt_discover_characteristics_rsp_t
{
    uint16 result
}
```

Table 3.166. Events Generated

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

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

3.9.1.2 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 3.167. 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, first byte is the length of UUID and rest is UUID in little-endian format |

Table 3.168. 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 |

BGScript command

```
call gatt_discover_characteristics_by_uuid(connection,service,uuid_len, uuid_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_discover_characteristics_by_uuid(uint8 connection, uint32 service, uint8array uuid);

/* Response id */
dumo_rsp_gatt_discover_characteristics_by_uuid_id

/* Response structure */
struct dumo_msg_gatt_discover_characteristics_by_uuid_rsp_t
{
    uint16 result
}
```

Table 3.169. Events Generated

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

3.9.1.3 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 successfully completed or failed with error.

Table 3.170. Command

| Byte | Type | Name | Description |
|------|--------|----------------|------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x03 | olen | 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 3.171. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_discover_descriptors(connection,characteristic)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_discover_descriptors(uint8 connection, uint16 characteristic);

/* Response id */
dumo_rsp_gatt_discover_descriptors_id

/* Response structure */
struct dumo_msg_gatt_discover_descriptors_rsp_t
{
    uint16 result
}
```

Table 3.172. Events Generated

| Event | Description |
|---------------------------------|--------------------------------------------------|
| gatt_descriptor | Discovered descriptor from remote GATT database. |

| Event | Description |
|---------------------------------------|-----------------------------------------------------------------|
| <code>gatt_procedure_completed</code> | Procedure has been successfully completed or failed with error. |

3.9.1.4 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 3.173. 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 3.174. 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 |

BGScript command

```
call gatt_discover_primary_services(connection)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_discover_primary_services(uint8 connection);

/* Response id */
dumo_rsp_gatt_discover_primary_services_id

/* Response structure */
struct dumo_msg_gatt_discover_primary_services_rsp_t
{
    uint16 result
}
```

Table 3.175. Events Generated

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

3.9.1.5 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 successfully completed or failed with error.

Table 3.176. Command

| Byte | Type | Name | Description |
|------|------------|------------|------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5 | uint8array | uuid | Characteristic UUID, first byte is the length of UUID and rest is UUID in little-endian format |

Table 3.177. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_discover_primary_services_by_uuid(connection,uuid_len, uuid_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_discover_primary_services_by_uuid(uint8 connection, uint8array uuid);

/* Response id */
dumo_rsp_gatt_discover_primary_services_by_uuid_id

/* Response structure */
struct dumo_msg_gatt_discover_primary_services_by_uuid_rsp_t
{
    uint16 result
}
```

Table 3.178. Events Generated

| Event | Description |
|------------------------------|-----------------------------------------------|
| gatt_service | Discovered service from remote GATT database. |

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

3.9.1.6 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 3.179. Command

| Byte | Type | Name | Description |
|------|-------|------------|------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | olen | 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 3.180. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_execute_characteristic_value_write(connection, flags)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_execute_characteristic_value_write(uint8 connection, uint8 flags);

/* Response id */
dumo_rsp_gatt_execute_characteristic_value_write_id

/* Response structure */
struct dumo_msg_gatt_execute_characteristic_value_write_rsp_t
{
    uint16 result
}
```

Table 3.181. Events Generated

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

3.9.1.7 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 3.182. Command

| Byte | Type | Name | Description |
|------|--------|------------|---------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | olen | 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 3.183. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_find_included_services(connection,service)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_find_included_services(uint8 connection, uint32 service);

/* Response id */
dumo_rsp_gatt_find_included_services_id

/* Response structure */
struct dumo_msg_gatt_find_included_services_rsp_t
{
    uint16 result
}
```

Table 3.184. Events Generated

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

3.9.1.8 cmd_gatt_find_information_request

This command is used to obtain the mapping of attribute handles with their associated types (UUIDs) in a given handle range.

Table 3.185. 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 | 0x12 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | start_handle | First requested handle number |
| 7-8 | uint16 | end_handle | Last requested handle number |

Table 3.186. 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 |

BGScript command

```
call gatt_find_information_request(connection,start_handle,end_handle)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_find_information_request(uint8 connection, uint16 start_handle, uint16 end_handle);

/* Response id */
dumo_rsp_gatt_find_information_request_id

/* Response structure */
struct dumo_msg_gatt_find_information_request_rsp_t
{
    uint16 result
}
```

Table 3.187. Events Generated

| Event | Description |
|---------------------------------------------|------------------------------------------------------------------------------------------|
| gatt_find_information_found | One or more attributes have been found. |
| gatt_procedure_completed | Find Information Request Transaction has successfully completed or failed with an error. |

3.9.1.9 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. 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. A received [evt_gatt_characteristic_value](#) event can be used to verify that the data has been transmitted. 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 3.188. 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 3.189. 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 | 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 |

BGScript command

```
call gatt_prepare_characteristic_value_write(connection,characteristic,offset,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_prepare_characteristic_value_write(uint8 connection, uint16 characteristic, uint16 offset,
uint8array value);

/* Response id */
dumo_rsp_gatt_prepare_characteristic_value_write_id

/* Response structure */
struct dumo_msg_gatt_prepare_characteristic_value_write_rsp_t
{
```

```
    uint16 result  
}
```

Table 3.190. Events Generated

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

3.9.1.10 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 length of the characteristic value returned by the remote GATT server is less than or equal to the size of the GATT MTU. If the length of the value exceeds the size of the GATT MTU more than one `gatt_characteristic_value` event is 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 3.191. Command

| Byte | Type | Name | Description |
|------|--------|----------------|-------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x03 | olen | 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 <code>gatt_characteristic</code> event |

Table 3.192. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_read_characteristic_value(connection,characteristic)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_read_characteristic_value(uint8 connection, uint16 characteristic);

/* Response id */
dumo_rsp_gatt_read_characteristic_value_id

/* Response structure */
struct dumo_msg_gatt_read_characteristic_value_rsp_t
{
    uint16 result
}
```

Table 3.193. 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. |

3.9.1.11 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 length of the characteristic value returned by the remote GATT server is less than or equal to the size of the GATT MTU. If the length of the value exceeds the size of the GATT MTU more than one [gatt_characteristic_value](#) event is 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 3.194. Command

| Byte | Type | Name | Description |
|------|------------|------------|---------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | olen | 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 3.195. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_read_characteristic_value_by_uuid(connection,service,uuid_len, uuid_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_read_characteristic_value_by_uuid(uint8 connection, uint32 service, uint8array uuid);

/* Response id */
dumo_rsp_gatt_read_characteristic_value_by_uuid_id

/* Response structure */
struct dumo_msg_gatt_read_characteristic_value_by_uuid_rsp_t
{
    uint16 result
}
```

Table 3.196. 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. |

3.9.1.12 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 length of the descriptor value returned by the remote GATT server is less than or equal to the size of the GATT MTU. If the length of the value exceeds the size of the GATT MTU more than one `gatt_descriptor_value` event is 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 3.197. Command

| Byte | Type | Name | Description |
|------|--------|------------|--------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0xe0 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | descriptor | The handle of the GATT characteristic descriptor |

Table 3.198. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0xe0 | method | Message ID |
| 4-5 | uint16 | result | Result code • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_read_descriptor_value(connection,descriptor)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_read_descriptor_value(uint8 connection, uint16 descriptor);

/* Response id */
dumo_rsp_gatt_read_descriptor_value_id

/* Response structure */
struct dumo_msg_gatt_read_descriptor_value_rsp_t
{
    uint16 result
}
```

Table 3.199. Events Generated

| Event | Description |
|---------------------------------------|-----------------------------------------------------------------|
| <code>gatt_descriptor_value</code> | Descriptor value received from the remote GATT server. |
| <code>gatt_procedure_completed</code> | Procedure has been successfully completed or failed with error. |

3.9.1.13 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 3.200. 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 3.201. 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 |

BGScript command

```
call gatt_read_multiple_characteristic_values(connection,characteristic_list_len,    characteristic_list_data)
(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_read_multiple_characteristic_values(uint8 connection, uint8array characteristic_list);

/* Response id */
dumo_rsp_gatt_read_multiple_characteristic_values_id

/* Response structure */
struct dumo_msg_gatt_read_multiple_characteristic_values_rsp_t
{
    uint16 result
}
```

Table 3.202. 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. |

3.9.1.14 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 (0x1e) 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 3.203. Command

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

Table 3.204. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0xd | method | Message ID |
| 4-5 | uint16 | result | Result code • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_send_characteristic_confirmation(connection)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_send_characteristic_confirmation(uint8 connection);

/* Response id */
dumo_rsp_gatt_send_characteristic_confirmation_id

/* Response structure */
struct dumo_msg_gatt_send_characteristic_confirmation_rsp_t
{
    uint16 result
}
```

3.9.1.15 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 it has failed with an error. The alternative way to subscribe attributes notification or subscribe attributes indication is to discover Client Characteristic Configuration Descriptor (with [cmd_gatt_find_information_request](#)), and then to write [expected value](#) (with [cmd_gatt_write_descriptor_value](#)) to its handler.

Table 3.205. Command

| Byte | Type | Name | Description |
|------|--------|----------------|----------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x04 | olen | 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 3.206. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_set_characteristic_notification(connection,characteristic,flags)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_set_characteristic_notification(uint8 connection, uint16 characteristic, uint8 flags);

/* Response id */
dumo_rsp_gatt_set_characteristic_notification_id

/* Response structure */
struct dumo_msg_gatt_set_characteristic_notification_rsp_t
{
    uint16 result
}
```

Table 3.207. Events Generated

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

3.9.1.16 cmd_gatt_set_max_mtu

This command can be used to set the maximum number of GATT Message Transfer Units (MTU). If max_mtu is non-default, MTU is exchanged automatically after Bluetooth LE connection has been established.

Table 3.208. Command

| Byte | Type | Name | Description |
|------|--------|---------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | max_mtu | Maximum number of Message Transfer Units (MTU) allowed <ul style="list-style-type: none"> Range: 23 to 158 Default: 23 |

Table 3.209. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> 0: success Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_set_max_mtu(max_mtu)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_set_max_mtu(uint16 max_mtu);

/* Response id */
dumo_rsp_gatt_set_max_mtu_id

/* Response structure */
struct dumo_msg_gatt_set_max_mtu_rsp_t
{
  uint16 result
}
```

3.9.1.17 cmd_gatt_write_characteristic_value

This command can be used to write the value of a characteristic in a remote GATT database. If the length of the given value is greater than the exchanged GATT MTU (Message Transfer Unit), "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 3.210. Command

| Byte | Type | Name | Description |
|------|------------|----------------|----------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | olen | 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 3.211. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_write_characteristic_value(connection,characteristic,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_write_characteristic_value(uint8 connection, uint16 characteristic, uint8array value);

/* Response id */
dumo_rsp_gatt_write_characteristic_value_id

/* Response structure */
struct dumo_msg_gatt_write_characteristic_value_rsp_t
{
    uint16 result
}
```

Table 3.212. Events Generated

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

3.9.1.18 cmd_gatt_write_characteristic_value_without_response

This command can be used to write the value of a characteristic in a remote GATT database. 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.

Table 3.213. 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 3.214. 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 | 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 |

BGScript command

```
call gatt_write_characteristic_value_without_response(connection,characteristic,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_write_characteristic_value_without_response(uint8 connection, uint16 characteristic,
uint8array value);

/* Response id */
dumo_rsp_gatt_write_characteristic_value_without_response_id

/* Response structure */
struct dumo_msg_gatt_write_characteristic_value_without_response_rsp_t
{
    uint16 result
}
```

3.9.1.19 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 length of the given value is greater than the exchanged GATT MTU size, "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 3.215. Command

| Byte | Type | Name | Description |
|------|------------|------------|--------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x04 | olen | 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 | The handle of the GATT characteristic descriptor |
| 7 | uint8array | value | Descriptor value |

Table 3.216. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_write_descriptor_value(connection,descriptor,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_write_descriptor_value(uint8 connection, uint16 descriptor, uint8array value);

/* Response id */
dumo_rsp_gatt_write_descriptor_value_id

/* Response structure */
struct dumo_msg_gatt_write_descriptor_value_rsp_t
{
    uint16 result
}
```

Table 3.217. Events Generated

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

3.9.2 gatt events

3.9.2.1 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 [gatt_discover_characteristics_by_uuid](#) command.

Table 3.218. Event

| Byte | Type | Name | Description |
|------|------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x05 | olen | 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 as a bitmask: • 0x01 : broadcast permitted • 0x02 : read permitted • 0x04 : write without response permitted • 0x08 : write permitted • 0x10 : notifications permitted • 0x20 : indications permitted • 0x40 : signed writes permitted • 0x80 : characteristic has extended properties Consult the GATT specification for more details. |
| 8 | uint8array | uuid | Characteristic UUID, first byte is the length of UUID and rest is UUID in little-endian format |

BGScript event

```
event gatt_characteristic(connection,characteristic,properties,uuid_len, uuid_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_characteristic_id

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

3.9.2.2 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_characteristic_value_by_uuid](#), [gatt_read_multiple_characteristic_values](#), [gatt_prepare_characteristic_value_write](#); 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 3.219. Event

| Byte | Type | Name | Description |
|------|------------|-------------------------|----------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x07 | olen | 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 | <code>att_opcode</code> | Attribute opcode which informs the GATT transaction used |
| 8-9 | uint16 | offset | Value offset |
| 10 | uint8array | value | Characteristic value |

BGScript event

```
event gatt_characteristic_value(connection,characteristic,att_opcode,offset,value_len, value_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_characteristic_value_id

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

3.9.2.3 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 3.220. Event

| Byte | Type | Name | Description |
|------|------------|------------|------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x04 | olen | 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 | The handle of the GATT characteristic descriptor |
| 7 | uint8array | uuid | Characteristic UUID, first byte is the length of UUID and rest is UUID in little-endian format |

BGScript event

```
event gatt_descriptor(connection,descriptor,uuid_len, uuid_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_descriptor_id

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

3.9.2.4 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 3.221. Event

| Byte | Type | Name | Description |
|------|------------|------------|--------------------------------------------------|
| 0 | 0xa0 | hlen | 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 | The handle of the GATT characteristic descriptor |
| 7-8 | uint16 | offset | Value offset |
| 9 | uint8array | value | Descriptor value |

BGScript event

```
event gatt_descriptor_value(connection,descriptor,offset,value_len, value_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_descriptor_value_id

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

3.9.2.5 evt_gatt_find_information_found

These events are generated when Find Information Response has been received (after `dumo_cmd_gatt_find_information_request` command execution). For ranges which contain more than one attribute, one event is generated for every attribute that was found. At the end of the event(s) sequence (after last Find Information Found event) a `dumo_evt_gatt_procedure_completed` event (with or without error code) will be generated.

Table 3.222. Event

| Byte | Type | Name | Description |
|------|------------|------------|------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x04 | olen | Minimum payload length |
| 2 | 0x09 | class | Message class: Generic Attribute Profile |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-6 | uint16 | handle | Attribute handle |
| 7 | uint8array | uuid | Attribute type (UUID) |

BGScript event

```
event gatt_find_information_found(connection,handle,uuid_len, uuid_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_find_information_found_id

/* Event structure */
struct dumo_msg_gatt_find_information_found_evt_t
{
    uint8 connection,
    uint16 handle,
    uint8array uuid
}
```

3.9.2.6 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 3.223. 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 |

BGScript event

```
event gatt_procedure_completed(connection,result)
```

C Functions

```
/* Event id */
dumo_evt_gatt_procedure_completed_id

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

3.9.2.7 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 [gatt_discover_primary_services_by_uuid](#) command.

Table 3.224. Event

| Byte | Type | Name | Description |
|------|------------|------------|------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x06 | olen | 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 | Characteristic UUID, first byte is the length of UUID and rest is UUID in little-endian format |

BGScript event

```
event gatt_service(connection,service,uuid_len, uuid_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_service_id

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

3.9.3 gatt enumerations

3.9.3.1 enum_gatt_att_opcode

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

Table 3.225. 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 |

3.9.3.2 enum_gatt_client_config_flag

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

Table 3.226. Enumerations

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

3.9.3.3 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 3.227. Enumerations

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

3.10 Generic Attribute Profile Server (gatt_server)

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

3.10.1 gatt_server commands

3.10.1.1 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 3.228. Command

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

Table 3.229. Response

| Byte | Type | Name | Description |
|------|------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8array | type | Variable length byte array. First byte is the length of the array. |

BGScript command

```
call gatt_server_read_attribute_type(attribute)(result,type_len, type_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_read_attribute_type(uint16 attribute);

/* Response id */
dumo_rsp_gatt_server_read_attribute_type_id

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

3.10.1.2 cmd_gatt_server_read_attribute_value

This command can be used to read the value of an attribute from a local GATT database.

Table 3.230. 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 3.231. 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 <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8array | value | Variable length byte array. First byte is the length of the array. |

BGScript command

```
call gatt_server_read_attribute_value(attribute,offset)(result,value_len, value_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_read_attribute_value(uint16 attribute, uint16 offset);

/* Response id */
dumo_rsp_gatt_server_read_attribute_value_id

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

3.10.1.3 cmd_gatt_server_send_characteristic_notification

This command can be used to send notifications or indications to a remote GATT client. Notification or indication is sent only if the client has enabled them by setting the corresponding flag to the Client Characteristic Configuration descriptor. A new indication cannot be sent before a confirmation from the GATT client is first received. The confirmation is indicated by [gatt_server_characteristic_status](#) event.

Table 3.232. Command

| Byte | Type | Name | Description |
|------|------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | olen | 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: <ul style="list-style-type: none">• 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 3.233. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_server_send_characteristic_notification(connection,characteristic,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_send_characteristic_notification(uint8 connection, uint16 characteristic, uint8array value);

/* Response id */
dumo_rsp_gatt_server_send_characteristic_notification_id

/* Response structure */
struct dumo_msg_gatt_server_send_characteristic_notification_rsp_t
{
    uint16 result
}
```

3.10.1.4 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.

Table 3.234. Command

| Byte | Type | Name | Description |
|------|------------|----------------|------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x05 | olen | 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. Values: • 0: No error • Non-zero: see error codes |
| 8 | uint8array | value | Characteristic value to send to the GATT client. Ignored if att_errorcode is not 0. |

Table 3.235. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_server_send_user_read_response(connection,characteristic,att_errorcode,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_send_user_read_response(uint8 connection, uint16 characteristic, uint8
att_errorcode, uint8array value);

/* Response id */
dumo_rsp_gatt_server_send_user_read_response_id

/* Response structure */
struct dumo_msg_gatt_server_send_user_read_response_rsp_t
{
```

```
    uint16 result  
}
```

3.10.1.5 cmd_gatt_server_send_user_write_response

This command must be used to send a response to a `gatt_server_user_write_request` event. The response needs to be sent within 30 seconds, otherwise no more GATT transactions are allowed by the remote side. If `attr_errorcode` is set to 0 the 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 3.236. 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 <code>gatt_characteristic</code> event |
| 7 | uint8 | att_errorcode | Attribute protocol error code. Values: <ul style="list-style-type: none"> • 0: No error • Non-zero: see error codes |

Table 3.237. 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 <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call gatt_server_send_user_write_response(connection,characteristic,att_errorcode)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_send_user_write_response(uint8 connection, uint16 characteristic, uint8 att_errorcode);

/* Response id */
dumo_rsp_gatt_server_send_user_write_response_id

/* Response structure */
struct dumo_msg_gatt_server_send_user_write_response_rsp_t
{
    uint16 result
}
```

3.10.1.6 cmd_gatt_server_set_service_status

This command can be used to enable or disable service visibility when other device perform Service Discovery procedure. Refer to 'constants' file produced by build process to find handles for services. Only services with 'id' attribute in gatt description file are stored in 'constants' file.

Table 3.238. Command

| Byte | Type | Name | Description |
|------|--------|---------|---------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x0a | class | Message class: Generic Attribute Profile Server |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | service | Service handle |
| 6 | uint8 | enable | Service enable flag. Values: • 0: Disable • 1: Enable |

Table 3.239. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_server_set_service_status(service,enable)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_set_service_status(uint16 service, uint8 enable);

/* Response id */
dumo_rsp_gatt_server_set_service_status_id

/* Response structure */
struct dumo_msg_gatt_server_set_service_status_rsp_t
{
    uint16 result
}
```

3.10.1.7 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 3.240. Command

| Byte | Type | Name | Description |
|------|------------|-----------|-------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x05 | olen | 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 3.241. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call gatt_server_write_attribute_value(attribute,offset,value_len, value_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_gatt_server_write_attribute_value(uint16 attribute, uint16 offset, uint8array value);

/* Response id */
dumo_rsp_gatt_server_write_attribute_value_id

/* Response structure */
struct dumo_msg_gatt_server_write_attribute_value_rsp_t
{
    uint16 result
}
```

3.10.2 gatt_server events

3.10.2.1 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 3.242. Event

| Byte | Type | Name | Description |
|------|------------|------------|------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x07 | olen | 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 |

BGScript event

```
event gatt_server_attribute_value(connection,attribute,att_opcode,offset,value_len, value_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_server_attribute_value_id

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

3.10.2.2 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 3.243. Event

| Byte | Type | Name | Description |
|------|--------|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x06 | olen | 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. |

BGScript event

```
event gatt_server_characteristic_status(connection,characteristic,status\_flags,client\_config\_flags)
```

C Functions

```
/* Event id */
dumo_evt_gatt_server_characteristic_status_id

/* Event structure */
struct dumo_msg_gatt_server_characteristic_status_evt_t
{
    uint8 connection,
    uint16 characteristic,
    uint8 status\_flags,
    uint16 client\_config\_flags
}
```

3.10.2.3 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 3.244. Event

| Byte | Type | Name | Description |
|------|--------|----------------|------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x06 | olen | Minimum payload length |
| 2 | 0xa0 | 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 |

BGScript event

```
event gatt_server_user_read_request(connection,characteristic,att_opcode,offset)
```

C Functions

```
/* Event id */
dumo_evt_gatt_server_user_read_request_id

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

3.10.2.4 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. 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.

Table 3.245. Event

| Byte | Type | Name | Description |
|------|------------|----------------|------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x07 | olen | 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 |

BGScript event

```
event gatt_server_user_write_request(connection,characteristic,att_opcode,offset,value_len, value_data)
```

C Functions

```
/* Event id */
dumo_evt_gatt_server_user_write_request_id

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

3.10.3 gatt_server enumerations

3.10.3.1 enum_gatt_server_characteristic_status_flag

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

Table 3.246. Enumerations

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

3.11 Hardware (hardware)

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

3.11.1 hardware commands

3.11.1.1 cmd_hardware_configure_gpio

This command can be used to configure the mode of an I/O port. After a boot, the module uses the default settings defined in hardware.xml, and this command can be used to override the default settings. Note that GPIO configurations set with this command do not persist over a reset. The hardware_gpio_output mode configuration sets the output pins in the push-pull mode.

Table 3.247. Command

| Byte | Type | Name | Description |
|------|--------|--------|---------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | port | Port index, where A=0, B=1. |
| 5-6 | uint16 | gpio | Index of the GPIO pin on the port which this command affects. |
| 7 | uint8 | mode | Pin mode |
| 8 | uint8 | pullup | Input pin configuration. |

Table 3.248. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 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 |

BGScript command

```
call hardware_configure_gpio(port, gpio, mode, pullup) (result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_configure_gpio(uint8 port, uint16 gpio, uint8 mode, uint8 pullup);

/* Response id */
dumo_rsp_hardware_configure_gpio_id

/* Response structure */
struct dumo_msg_hardware_configure_gpio_rsp_t
{
    uint16 result
}
```

3.11.1.2 cmd_hardware_read_adc

This command can be used to read the specified channel of the A/D converter in the module. Note that the ADC channels must be configured in the hardware.xml file, as described in the module Configuration Guide. Note that only channels 4 through 7 are connected, Data read from channels 0 to 3 will be undefined. ADC value of 4095 corresponds to the input being at Vdd, value of which can be obtained with [read vdd command](#)

Table 3.249. Command

| Byte | Type | Name | Description |
|------|-------|--------|--------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | input | ADC input channel. Value range: 4-7. |

Table 3.250. Response

| Byte | Type | Name | Description |
|------|--------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | input | ADC input channel. Value range: 4-7. |
| 7-8 | uint16 | value | ADC value |

BGScript command

```
call hardware_read_adc(input)(result,input,value)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_read_adc(uint8 input);

/* Response id */
dumo_rsp_hardware_read_adc_id

/* Response structure */
struct dumo_msg_hardware_read_adc_rsp_t
{
    uint16 result,
    uint8 input,
    uint16 value
}
```

3.11.1.3 cmd_hardware_read_gpio

This command can be used to read the pins of the specified I/O-port of the module.

Table 3.251. Command

| Byte | Type | Name | Description |
|------|--------|--------|--------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | port | Port index to read from, A=0, B=1. |
| 5-6 | uint16 | mask | Bitmask of which pins on the port should be read |

Table 3.252. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x03 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6-7 | uint16 | data | Port data |

BGScript command

```
call hardware_read_gpio(port,mask)(result,data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_read_gpio(uint8 port, uint16 mask);

/* Response id */
dumo_rsp.hardware_read_gpio_id

/* Response structure */
struct dumo_msg.hardware_read_gpio_rsp_t
{
    uint16 result,
    uint16 data
}
```

3.11.1.4 cmd_hardware_read_i2c

This command can be used to read from the specified I2C interface. The I2C interfaces must be configured in the hardware.xml file, as described in the module Configuration Guide.

Table 3.253. Command

| Byte | Type | Name | Description |
|------|--------|---------------|--------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | channel | I2C channel to use. Value range: 0-1 |
| 5-6 | uint16 | slave_address | Slave address to use |
| 7 | uint8 | length | Amount of data to read |

Table 3.254. Response

| Byte | Type | Name | Description |
|------|------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 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 | uint8array | data | Data that was read if command was successful |

BGScript command

```
call hardware_read_i2c(channel,slave_address,length)(result,data_len, data_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_read_i2c(uint8 channel, uint16 slave_address, uint8 length);

/* Response id */
dumo_rsp_hardware_read_i2c_id

/* Response structure */
struct dumo_msg_hardware_read_i2c_rsp_t
{
    uint16 result,
    uint8array data
}
```

3.11.1.5 cmd_hardware_read_junction_temperature

This command can be used to read the junction temperature of the module's MCU. Command is active only if the ADC is active. For more information, see the Hardware Configuration Guide.

Table 3.255. Command

| Byte | Type | Name | Description |
|------|------|--------|-------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0xa0 | method | Message ID |

Table 3.256. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0xa0 | 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 | value | Junction temperature in Celsius |

BGScript command

```
call hardware_read_junction_temperature()(result,value)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_read_junction_temperature();

/* Response id */
dumo_rsp_hardware_read_junction_temperature_id

/* Response structure */
struct dumo_msg_hardware_read_junction_temperature_rsp_t
{
    uint16 result,
    uint16 value
}
```

3.11.1.6 cmd_hardware_read_vdd

This command can be used to read the voltage level of the Vdd pin. This command is active only if the ADC is active. For more information see the Hardware Configuration Guide.

Table 3.257. Command

| Byte | Type | Name | Description |
|------|------|--------|-------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x09 | method | Message ID |

Table 3.258. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 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 |
| 6-7 | uint16 | value | Voltage of Vdd in millivolts |

BGScript command

```
call hardware_read_vdd()(result,value)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_read_vdd();

/* Response id */
dumo_rsp_hardware_read_vdd_id

/* Response structure */
struct dumo_msg_hardware_read_vdd_rsp_t
{
    uint16 result,
    uint16 value
}
```

3.11.1.7 cmd_hardware_read_write_spi

This command can be used to read from and write to the specified SPI interface. The SPI interfaces must be configured in the hardware.xml file, as described in the module Configuration Guide. SPI slave is working in the blocking mode, waiting for CS signal and clock from master.

Table 3.259. Command

| Byte | Type | Name | Description |
|------|------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0b | method | Message ID |
| 4 | uint8 | channel | SPI channel to use. Value range: 1-2 |
| 5 | uint8 | cs_port | Chip select pin port. Value range: 0 (A) or 1 (B) |
| 6 | uint8 | cs_pin | Chip select pin. Value range: 1-15 or 0 to disable (output in master mode , input in slave mode). |
| 7 | uint8 | cs_polarity | Chip select pin polarity. |
| 8 | uint8 | read_length | Amount of data to read and write. Note: SPI read and write are synchronous and you will receive at least as many bytes as you have written. If you reading more than writing then the writing will be zero padded to match the length of the data to read. |
| 9 | uint8array | data | Data to write |

Table 3.260. Response

| Byte | Type | Name | Description |
|------|------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0b | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8array | data | Data that was read if command was successful |

BGScript command

```
call      hardware_read_write_spi(channel,cs_port,cs_pin,cs_polarity,read_length,data_len,           data_data)
(result,data_len, data_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_read_write_spi(uint8 channel, uint8 cs_port, uint8 cs_pin, uint8 cs_polarity, uint8
read_length, uint8array data);

/* Response id */
dumo_rsp_hardware_read_write_spi_id
```

```
/* Response structure */
struct dumo_msg_hardware_read_write_spi_rsp_t
{
    uint16 result,
    uint8array data
}
```

3.11.1.8 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 ID's available, but in practice the amount of concurrent timers is limited by the amount of free memory.

Table 3.261. Command

| Byte | Type | Name | Description |
|------|--------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x06 | olen | 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 milliseconds. If time is 0, removes the scheduled timer. The maximum value is 4294967295, which corresponds to about 49.7 days. |
| 8 | uint8 | timer_id | Handle that is returned with event |
| 9 | uint8 | single_shot | Timer mode. Values: <ul style="list-style-type: none">• 0: false (timer is repeating)• 1: true (timer runs only once) |

Table 3.262. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call hardware_set_soft_timer(time,timer_id,single_shot)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_set_soft_timer(uint32 time, uint8 timer_id, uint8 single_shot);

/* Response id */
dumo_rsp_hardware_set_soft_timer_id

/* Response structure */
struct dumo_msg_hardware_set_soft_timer_rsp_t
{
    uint16 result
}
```

Table 3.263. Events Generated

| Event | Description |
|-------------------------------------|-------------------------------|
| hardware_soft_timer | Sent after specified interval |

3.11.1.9 cmd_hardware_set_uart_bgapi_mode

This command can be used to switch the UART interface between BGAPI and DATA mode. In BGAPI mode all data frames defined in BGAPI protocol specification are processed (and also generated) by the firmware. In DATA mode all UART's data are routed "from" and "to" selected endpoint as RAW data.

Table 3.264. Command

| Byte | Type | Name | Description |
|------|-------|------------|--------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x0c | method | Message ID |
| 4 | uint8 | bgapi_mode | Values: • 1 : Set UART in BGAPI mode • 0 : Set UART in DATA mode |

Table 3.265. 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 |

BGScript command

```
call hardware_set_uart_bgapi_mode(bgapi_mode)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_set_uart_bgapi_mode(uint8 bgapi_mode);

/* Response id */
dumo_rsp_hardware_set_uart_bgapi_mode_id

/* Response structure */
struct dumo_msg_hardware_set_uart_bgapi_mode_rsp_t
{
    uint16 result
}
```

3.11.1.10 cmd_hardware_set_uart_configuration

This command can be used to reconfigure the UART interface. The new configuration will become effective after a 100 ms delay from issuing the command. The response to the command will be sent before the 100 ms delay has elapsed using the original UART parameters. Any new configuration parameter is lost after reset and the firmware default parameters will be used again.

Table 3.266. Command

| Byte | Type | Name | Description |
|------|--------|-----------|---------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint. Must be 0. |
| 5-8 | uint32 | rate | UART baud rate. Range: 9600 to 4000000 |
| 9 | uint8 | data_bits | Data bits. Must be 8. |
| 10 | uint8 | stop_bits | Stop bits. Values: <ul style="list-style-type: none">• 1: 1 stop bit• 2: 2 stop bits |
| 11 | uint8 | parity | Parity bit. Values: see link |
| 12 | uint8 | flow_ctrl | UART flow control. Values: <ul style="list-style-type: none">• 0: None• 1: RTS/CTS |

Table 3.267. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x08 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call hardware_set_uart_configuration(endpoint,rate,data_bits,stop_bits,parity,flow_ctrl)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_set_uart_configuration(uint8 endpoint, uint32 rate, uint8 data_bits, uint8 stop_bits,
uint8 parity, uint8 flow_ctrl);

/* Response id */
dumo_rsp_hardware_set_uart_configuration_id

/* Response structure */
```

```
struct dumo_msg_hardware_set_uart_configuration_rsp_t
{
    uint16 result
}
```

3.11.1.11 cmd_hardware_stop_i2c

This command can be used to stop I2C transmission.

Table 3.268. Command

| Byte | Type | Name | Description |
|------|-------|---------|-------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | channel | I2C channel to use. |

Table 3.269. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 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 |

BGScript command

```
call hardware_stop_i2c(channel)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_stop_i2c(uint8 channel);

/* Response id */
dumo_rsp_hardware_stop_i2c_id

/* Response structure */
struct dumo_msg_hardware_stop_i2c_rsp_t
{
    uint16 result
}
```

3.11.1.12 cmd_hardware_write_gpio

This command can be used to set the logic states of pins of the specified I/O-port using a bitmask. Correct usage of this command requires output pins configuration with the [cmd_hardware_configure_gpio](#) (or port configuration in hardware configuration file) at first.

Table 3.270. Command

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | port | Port index, where A=0, B=1. |
| 5-6 | uint16 | mask | Bitmask which determines the pins this command is used to set |
| 7-8 | uint16 | data | Bitmask of which pins to set high and which pins to set low (1 is high, 0 is low) |

Table 3.271. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call hardware_write_gpio(port,mask,data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_write_gpio(uint8 port, uint16 mask, uint16 data);

/* Response id */
dumo_rsp.hardware_write_gpio_id

/* Response structure */
struct dumo_msg.hardware_write_gpio_rsp_t
{
    uint16 result
}
```

3.11.1.13 cmd_hardware_write_i2c

This command can be used to write data into I2C interface. The I2C interfaces must be configured in the hardware.xml file, as described in the module Configuration Guide.

Table 3.272. Command

| Byte | Type | Name | Description |
|------|------------|---------------|--------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | channel | I2C channel to use. Value range: 0-1 |
| 5-6 | uint16 | slave_address | Slave address to use |
| 7 | uint8array | data | Data to write |

Table 3.273. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 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 |

BGScript command

```
call hardware_write_i2c(channel,slave_address,data_len, data_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_hardware_write_i2c(uint8 channel, uint16 slave_address, uint8array data);

/* Response id */
dumo_rsp_hardware_write_i2c_id

/* Response structure */
struct dumo_msg_hardware_write_i2c_rsp_t
{
    uint16 result
}
```

3.11.2 hardware events

3.11.2.1 evt_hardware_interrupt

This event indicates that an external interrupt has occurred and provides a timestamp and a mask which indicates all triggered interrupt channels.

Table 3.274. Event

| Byte | Type | Name | Description |
|------|--------|------------|----------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x0c | class | Message class: Hardware |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | interrupts | Mask of interrupt channels |
| 6-9 | uint32 | timestamp | Timestamp |

BGScript event

```
event hardware_interrupt(interrupts,timestamp)
```

C Functions

```
/* Event id */
dumo_evt_hardware_interrupt_id

/* Event structure */
struct dumo_msg_hardware_interrupt_evt_t
{
    uint16 interrupts,
    uint32 timestamp
}
```

3.11.2.2 evt_hardware_soft_timer

This event indicates that the soft timer has lapsed.

Table 3.275. Event

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

BGScript event

```
event hardware_soft_timer(timer_id)
```

C Functions

```
/* Event id */
dumo_evt_hardware_soft_timer_id

/* Event structure */
struct dumo_msg_hardware_soft_timer_evt_t
{
    uint8 timer_id
}
```

3.11.3 hardware enumerations

3.11.3.1 enum_hardware_cspolarity

These values define the polarity of a Chip Select pin.

Table 3.276. Enumerations

| Value | Name | Description |
|-------|--------------------|------------------------------------------------------------------------------------|
| 0 | hardware_low | Active low (low during transmission, high during inactivity) |
| 1 | hardware_high | Active high (high during transmission, low during inactivity) |
| 2 | hardware_opendrain | Open drain (only in master mode, active low). Used with external pull-up resistor. |

3.11.3.2 enum_hardware_gpio_configuration

These values define whether the pin is used in pull-up, pull-down or in no pull-up and no pull-down mode. Pull-up and pull-down resistors are only available for input pins.

Table 3.277. Enumerations

| Value | Name | Description |
|-------|------------------------|--------------------------|
| 0 | hardware_gpio_float | No pull-up, no pull-down |
| 1 | hardware_gpio_pullup | Pull-up |
| 2 | hardware_gpio_pulldown | Pull-down |

3.11.3.3 enum_hardware_gpio_mode

These values define whether the pin is used as an input, an output, as a function pin or as an analog signal pin.

Table 3.278. Enumerations

| Value | Name | Description |
|-------|------------------------|----------------------------------------------|
| 0 | hardware_gpio_input | Input |
| 1 | hardware_gpio_output | Output (the push-pull digital output driver) |
| 2 | hardware_gpio_function | Function |
| 3 | hardware_gpio_analog | Analog |

3.11.3.4 enum_hardware_uartparity

These values define the parity bit configuration of the related UART connection.

Table 3.279. Enumerations

| Value | Name | Description |
|-------|---------------|-------------|
| 0 | hardware_none | None |
| 1 | hardware_odd | Odd parity |
| 2 | hardware_even | Even parity |

3.12 Identity Profile (identity)

The commands and events in this class are related to Bluetooth Classic Identity Profile operations.

3.12.1 identity commands

3.12.1.1 cmd_identity_local_identity

This command can be used to read Identity Profile information from the local device.

Table 3.280. Command

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity Profile |
| 3 | 0x01 | method | Message ID |

Table 3.281. Response

| Byte | Type | Name | Description |
|-------|------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x0b | lolen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity Profile |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6-7 | uint16 | source | The authority who has issued the Vendor ID. Values: <ul style="list-style-type: none"> • 1: Bluetooth SIG • 0x0002: USB IF |
| 8-9 | uint16 | vendor | Vendor ID |
| 10-11 | uint16 | product | Product ID |
| 12-13 | uint16 | version | Version ID |
| 14 | uint8array | description | Service Description |

BGScript command

```
call identity_local_identity()(result,source,vendor,product,version,description_len, description_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_identity_local_identity();

/* Response id */
dumo_rsp_identity_local_identity_id

/* Response structure */
struct dumo_msg_identity_local_identity_rsp_t
{
    uint16 result,
    uint16 source,
    uint16 vendor,
    uint16 product,
    uint16 version,
```

```
    uint8array description
}
```

3.12.1.2 cmd_identity_modify_local_identity

This command can be used to modify some fields in the local identity profile.

Table 3.282. Command

| Byte | Type | Name | Description |
|------|--------|---------|---------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x06 | olen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity Profile |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | vendor | Vendor ID. |
| 6-7 | uint16 | product | Product ID |
| 8-9 | uint16 | version | Version ID |

Table 3.283. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity 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 |

BGScript command

```
call identity_modify_local_identity(vendor,product,version)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_identity_modify_local_identity(uint16 vendor, uint16 product, uint16 version);

/* Response id */
dumo_rsp_identity_modify_local_identity_id

/* Response structure */
struct dumo_msg_identity_modify_local_identity_rsp_t
{
    uint16 result
}
```

3.12.1.3 cmd_identity_remote_identity

This command can be used to read Identity Profile information from a remote Bluetooth Classic device.

Table 3.284. Command

| Byte | Type | Name | Description |
|------|---------|---------|---------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity Profile |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth Classic address in little endian format |

Table 3.285. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity 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 |

BGScript command

```
call identity_remote_identity(address)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_identity_remote_identity(bd_addr *address);

/* Response id */
dumo_rsp_identity_remote_identity_id

/* Response structure */
struct dumo_msg_identity_remote_identity_rsp_t
{
    uint16 result
}
```

Table 3.286. Events Generated

| Event | Description |
|--------------------------|------------------------------------------------------------------------------------------------------------|
| identity_remote_identity | This event indicates the reception of Identity Profile information from a remote Bluetooth Classic device. |

3.12.2 identity events

3.12.2.1 evt_identity_remote_identity

This event indicates the reception of Identity Profile information from a remote Bluetooth Classic device.

Table 3.287. Event

| Byte | Type | Name | Description |
|-------|------------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x11 | olen | Minimum payload length |
| 2 | 0x05 | class | Message class: Identity Profile |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | status | Result code • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6-11 | bd_addr | address | Bluetooth address of remote device in little endian format |
| 12-13 | uint16 | source | The authority who has issued the Vendor ID. Values: • 1: Bluetooth SIG • 0x0002: USB IF |
| 14-15 | uint16 | vendor | Vendor ID. A comprehensive list of Bluetooth SIG -assigned ID's is at https://www.bluetooth.org/en-us/specification/assigned-numbers/company-identifiers |
| 16-17 | uint16 | product | Product ID |
| 18-19 | uint16 | version | Version ID |
| 20 | uint8array | description | Service Description |

BGScript event

```
event identity_remote_identity(status,address,source, vendor, product, version, description_len, description_data)
```

C Functions

```
/* Event id */
dumo_evt_identity_remote_identity_id

/* Event structure */
struct dumo_msg_identity_remote_identity_evt_t
{
    uint16 status,
    bd_addr *address,
    uint16 source,
    uint16 vendor,
    uint16 product,
    uint16 version,
    uint8array description
}
```

3.13 Connection management for Bluetooth Low Energy (`le_connection`)

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

3.13.1 `le_connection` commands

3.13.1.1 cmd_le_connection_list

This command can be used to list all LE connections and check their parameters.

Table 3.288. Command

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 3 | 0x01 | method | Message ID |

Table 3.289. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_connection_list()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_connection_list();

/* Response id */
dumo_rsp_le_connection_list_id

/* Response structure */
struct dumo_msg_le_connection_list_rsp_t
{
    uint16 result
}
```

Table 3.290. Events Generated

| Event | Description |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| le_connection_parameters | This event is triggered whenever the connection parameters are changed and at any time a connection is established. The event is also generated by LE connection listing. |
| le_connection_list_complete | This event indicates that all connections have been listed. |

3.13.1.2 cmd_le_connection_set_parameters

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

Table 3.291. Command

| Byte | Type | Name | Description |
|-------|--------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x09 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 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 less than or equal to max_interval. <ul style="list-style-type: none">• 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. <ul style="list-style-type: none">• 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 intervals the slave can skip if it has no data to send <ul style="list-style-type: none">• Range: 0x0000 to 0x01f4Use 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. <ul style="list-style-type: none">• Range: 0x000a to 0x0c80• Time = Value x 10 ms• Time Range: 100 ms to 32 s• The minimum value must be at least maximum interval in ms * (latency + 1) * 2 <p>It is recommended that the supervision timeout is set at a value which allows communication attempts over at least a few connection intervals.</p> |

Table 3.292. Response

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

| Byte | Type | Name | Description |
|------|--------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4-5 | uint16 | result | <p>Result code</p> <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred <p>For other values refer to the Error codes</p> |

BGScript command

```
call le_connection_set_parameters(connection,min_interval,max_interval,latency,timeout)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_connection_set_parameters(uint8 connection, uint16 min_interval, uint16 max_interval, uint16 latency, uint16 timeout);

/* Response id */
dumo_rsp_le_connection_set_parameters_id

/* Response structure */
struct dumo_msg_le_connection_set_parameters_rsp_t
{
    uint16 result
}
```

3.13.2 le_connection events

3.13.2.1 evt_le_connection_closed

This event indicates that a connection was closed.

Table 3.293. Event

| Byte | Type | Name | Description |
|------|--------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | reason | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | connection | Handle of the closed connection |

BGScript event

```
event le_connection_closed(reason,connection)
```

C Functions

```
/* Event id */
dumo_evt_le_connection_closed_id

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

3.13.2.2 evt_le_connection_list_complete

This event indicates that all connections have been listed.

Table 3.294. Event

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 3 | 0x03 | method | Message ID |

BGScript event

```
event le_connection_list_complete()
```

C Functions

```
/* Event id */
dumo_evt_le_connection_list_complete_id

/* Event structure */
struct dumo_msg_le_connection_list_complete_evt_t
{
}
```

3.13.2.3 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 module (Slave or Master). After entering the Connection State, the connection is considered to be created. The connection is not considered to be established at this point. A connection is only considered to be established once a data channel packet has been received from the peer device. An open connection can be closed with the command `dumo_cmd_endpoint_close` by giving the connection handle ID obtained from this event.

Table 3.295. Event

| Byte | Type | Name | Description |
|------|---------|--------------|----------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x0a | olen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Remote device address |
| 10 | uint8 | address_type | Remote device address type |
| 11 | uint8 | master | Module 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 |

BGScript event

```
event le_connection_opened(address,address_type, master, connection, bonding)
```

C Functions

```
/* Event id */
dumo_evt_le_connection_opened_id

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

3.13.2.4 evt_le_connection_parameters

This event is triggered whenever the connection parameters are changed and at any time a connection is established. The event is also generated by LE connection listing.

Table 3.296. Event

| Byte | Type | Name | Description |
|-------|---------|---------------|---------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x0e | olen | Minimum payload length |
| 2 | 0x08 | class | Message class: Connection management for Bluetooth Low Energy |
| 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 |
| 12-17 | bd_addr | address | Remote device address |

BGScript event

```
event le_connection_parameters(connection,interval,latency,timeout,security_mode,address)
```

C Functions

```
/* Event id */
dumo_evt_le_connection_parameters_id

/* Event structure */
struct dumo_msg_le_connection_parameters_evt_t
{
    uint8 connection,
    uint16 interval,
    uint16 latency,
    uint16 timeout,
    uint8 security_mode,
    bd_addr *address
}
```

3.13.3 le_connection enumerations

3.13.3.1 enum_le_connection_security

These values indicate the Bluetooth LE Security Mode.

Table 3.297. 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 |

3.14 Generic Access Profile, Bluetooth Low Energy (le_gap)

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

3.14.1 le_gap commands

3.14.1.1 cmd_le_gap_discover

This command can be used to start the GAP discovery procedure to scan for advertising devices, that is to perform a device discovery. Scanning parameters can be configured with the [le_gap_set_scan_parameters](#) command before issuing this command. To cancel an ongoing discovery process use the [le_gap_end_procedure](#) command.

Table 3.298. Command

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

Table 3.299. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_discover(mode)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_discover(uint8 mode);

/* Response id */
dumo_rsp_le_gap_discover_id

/* Response structure */
struct dumo_msg_le_gap_discover_rsp_t
{
    uint16 result
}
```

Table 3.300. Events Generated

| Event | Description |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| le_gap_scan_response | Every time an advertisement 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. |

3.14.1.2 cmd_le_gap_end_procedure

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

Table 3.301. 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, Bluetooth Low Energy |
| 3 | 0x03 | method | Message ID |

Table 3.302. 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, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_end_procedure()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_end_procedure();

/* Response id */
dumo_rsp_le_gap_end_procedure_id

/* Response structure */
struct dumo_msg_le_gap_end_procedure_rsp_t
{
    uint16 result
}
```

3.14.1.3 cmd_le_gap_open

This command can be used to open a Bluetooth LE connection. Scanning parameters can be configured with the le_gap_set_scan_parameters command before issuing this command. To cancel on an ongoing connection process use the [endpoint_close](#) command.

Table 3.303. Command

| Byte | Type | Name | Description |
|------|---------|--------------|-------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x07 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 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 3.304. Response

| Byte | Type | Name | Description |
|------|--------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 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 is established. This handle is valid only if the result code of this response is 0 (zero). |

BGScript command

```
call le_gap_open(address,address_type)(result,connection)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_open(bd_addr *address, uint8 address_type);

/* Response id */
dumo_rsp_le_gap_open_id

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

Table 3.305. 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 module (Slave or Master). |

3.14.1.4 cmd_le_gap_scan_filter_clear

This command can be used to remove all scan filters.

Table 3.306. Command

| Byte | Type | Name | Description |
|------|------|--------|-------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x00 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x0b | method | Message ID |

Table 3.307. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_scan_filter_clear()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_scan_filter_clear();

/* Response id */
dumo_rsp_le_gap_scan_filter_clear_id

/* Response structure */
struct dumo_msg_le_gap_scan_filter_clear_rsp_t
{
    uint16 result
}
```

3.14.1.5 cmd_le_gap_set_adv_data

This command can be used to set the data in advertisement packets or in the scan response packets. This data is used when advertising in user data mode. It is recommended to set both the advertisement data and scan response data at the same time.

Table 3.308. 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, Bluetooth Low Energy |
| 3 | 0x07 | method | Message ID |
| 4 | uint8 | scan_rsp | This value selects if the data is intended for advertisement packets or scan response packets. Values: <ul style="list-style-type: none"> • 0: Advertisement packets • 1: Scan response packets |
| 5 | uint8array | adv_data | Data to be set. Maximum length is 31 bytes which is the maximum user data that will fit in the payload of an advertising channel PDU, according to the specification. |

Table 3.309. 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, Bluetooth Low Energy |
| 3 | 0x07 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call le_gap_set_adv_data(scan_rsp,adv_data_len, adv_data_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_adv_data(uint8 scan_rsp, uint8array adv_data);

/* Response id */
dumo_rsp_le_gap_set_adv_data_id

/* Response structure */
struct dumo_msg_le_gap_set_adv_data_rsp_t
{
    uint16 result
}
```

3.14.1.6 cmd_le_gap_set_adv_parameters

This command can be used to set Bluetooth LE advertisement parameters.

Table 3.310. Command

| Byte | Type | Name | Description |
|------|--------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | interval_min | Minimum advertising intervalValue multiplied in units of 0.625 msFor connectable advertisingRange: 0x0020 to 0x4000Time range: 20 ms to 10.24 sFor non connectable advertisingRange: 0x00a0 to 0x4000Time range: 100 ms to 10.24 s default: 0xa0 |
| 6-7 | uint16 | interval_max | Maximum advertising interval. Value in units of 0.625 ms <ul style="list-style-type: none"> • Range: 0x0020 to 0x4000 • Time range: 20 ms to 10.24 s • Note: interval_max must be at least equal to or bigger than interval_min • default: 0x140 |
| 8 | 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: <ul style="list-style-type: none"> • 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/default value: 7 |

Table 3.311. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x04 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call le_gap_set_adv_parameters(interval_min,interval_max,channel_map)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_adv_parameters(uint16 interval_min, uint16 interval_max, uint8 channel_map);

/* Response id */
dumo_rsp_le_gap_set_adv_parameters_id

/* Response structure */
struct dumo_msg_le_gap_set_adv_parameters_rsp_t
{
    uint16 result
}
```

3.14.1.7 cmd_le_gap_set_conn_parameters

This command can be used to set the default Bluetooth LE 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 3.312. Command

| Byte | Type | Name | Description |
|-------|--------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x08 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | min_interval | <p>Minimum value for the connection event interval. This must be set less than or equal to max_interval.</p> <ul style="list-style-type: none"> Time = Value x 1.25 ms Range: 0x0006 to 0x0c80 Time Range: 7.5 ms to 4 s default: 100 |
| 6-7 | uint16 | max_interval | <p>Maximum value for the connection event interval. This must be set greater than or equal to min_interval.</p> <ul style="list-style-type: none"> Time = Value x 1.25 ms Range: 0x0006 to 0x0c80 Time Range: 7.5 ms to 4 s default: 200 |
| 8-9 | uint16 | latency | <p>Slave latency. This parameter defines how many connection intervals the slave can skip if it has no data to send</p> <ul style="list-style-type: none"> Range: 0x0000 to 0x01f4 <p>Use 0x0000 for default value</p> |
| 10-11 | uint16 | timeout | <p>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.</p> <ul style="list-style-type: none"> Range: 0x000a to 0x0c80 Time = Value x 10 ms Time Range: 100 ms to 32 s The minimum value must be at least maximum interval * (latency + 1) default: 100 <p>It is recommended that the supervision timeout is set at a value which allows communication attempts over at least a few connection intervals.</p> |

Table 3.313. Response

| Byte | Type | Name | Description |
|------|------|--------|-------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x05 | method | Message ID |

| Byte | Type | Name | Description |
|------|--------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4-5 | uint16 | result | <p>Result code</p> <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred <p>For other values refer to the Error codes</p> |

BGScript command

```
call le_gap_set_conn_parameters(min_interval,max_interval,latency,timeout)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_conn_parameters(uint16 min_interval, uint16 max_interval, uint16 latency, uint16 timeout);

/* Response id */
dumo_rsp_le_gap_set_conn_parameters_id

/* Response structure */
struct dumo_msg_le_gap_set_conn_parameters_rsp_t
{
    uint16 result
}
```

3.14.1.8 cmd_le_gap_set_host_channel_classification

This command configures Bluetooth LE channel classifications. If successful, a [bt_gap_host_channel_classification_complete](#) event will follow when the controller has completed the configuration.

Table 3.314. Command

| Byte | Type | Name | Description |
|------|------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x09 | method | Message ID |
| 4 | uint8array | channel_map | Channel bitmap for 37 channels. Mark known bad channels with a zero bit, other channels with a one bit. The parameter must be given as exactly 5 bytes; the 3 high bits are ignored. |

Table 3.315. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_set_host_channel_classification(channel_map_len, channel_map_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_host_channel_classification(uint8array channel_map);

/* Response id */
dumo_rsp_le_gap_set_host_channel_classification_id

/* Response structure */
struct dumo_msg_le_gap_set_host_channel_classification_rsp_t
{
    uint16 result
}
```

3.14.1.9 cmd_le_gap_set_max_power

This command can be used to set the maximum TX power for Bluetooth LE.

Table 3.316. 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, Bluetooth Low Energy |
| 3 | 0x08 | method | Message ID |
| 4 | int8 | max_power | Maximum radio TX power. Values: • 0 : 7dBm • 1 : 2dBm • 2 : -3dBm • 3 : -8 dBm • 4 : -13 dBm • 5 : -18 dBm • Default value: 0 |

Table 3.317. 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, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_set_max_power(max_power)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_max_power(int8 max_power);

/* Response id */
dumo_rsp_le_gap_set_max_power_id

/* Response structure */
struct dumo_msg_le_gap_set_max_power_rsp_t
{
    uint16 result
}
```

3.14.1.10 cmd_le_gap_set_mode

This command can be used to configure the current Bluetooth LE GAP Connectable and Discoverable modes. It can be used to enable advertisements and/or allow incoming connections. To exit from this mode (to stop advertising and/or disallow incoming connections), issue this command with the Not Discoverable / Not Connectable parameter values. After connection establishment with remote master module, advertising on the slave module is stopped and switched to non - connectable mode. Availability of the discoverable and connectable modes depend on device role (advertiser, scanner, master, slave). For more information see Bluetooth 4.0 specification.

Table 3.318. Command

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

Table 3.319. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_set_mode(discover,connect)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_mode(uint8 discover, uint8 connect);

/* Response id */
dumo_rsp_le_gap_set_mode_id

/* Response structure */
struct dumo_msg_le_gap_set_mode_rsp_t
{
    uint16 result
}
```

3.14.1.11 cmd_le_gap_set_random_static_address

This command can be used to set random static address for LE

Table 3.320. 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, Bluetooth Low Energy |
| 3 | 0x0c | method | Message ID |
| 4-9 | bd_addr | static_address | Bluetooth address |
| 10 | uint8 | enable | Unsigned 8-bit integer |

Table 3.321. 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, Bluetooth Low Energy |
| 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 |

BGScript command

```
call le_gap_set_random_static_address(static_address,enable)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_random_static_address(bd_addr *static_address, uint8 enable);

/* Response id */
dumo_rsp_le_gap_set_random_static_address_id

/* Response structure */
struct dumo_msg_le_gap_set_random_static_address_rsp_t
{
    uint16 result
}
```

3.14.1.12 cmd_le_gap_set_scan_parameters

This command can be used to set Bluetooth LE scan parameters.

Table 3.322. 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, Bluetooth Low Energy |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | scan_interval | Scanner interval. This is defined as the time interval from when the module started its last LE scan until it begins the subsequent LE scan, that is how often to scan <ul style="list-style-type: none"> • Time = Value x 0.625 ms • Range: 0x0004 to 0x4000 • Time Range: 2.5 ms to 10.24 s • Default: 0x0010 (10 ms) |
| 6-7 | uint16 | scan_window | Scan window. The duration of the LE scan. scan_window shall be less than or equal to scan_interval <ul style="list-style-type: none"> • Time = Value x 0.625 ms • Range: 0x0004 to 0x4000 • Time Range: 2.5 ms to 10.24 s • Default: 0x0010 (10 ms) |
| 8 | uint8 | active | Scan type indicated by a flag. Values: <ul style="list-style-type: none"> • 0: Passive scanning • 1: Active scanning • Default value: 0 • In passive scanning mode the module only listens to advertising packets and will not transmit any packet • In active scanning mode the module 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 |

Table 3.323. 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, Bluetooth Low Energy |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call le_gap_set_scan_parameters(scan_interval,scan_window,active)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_scan_parameters(uint16 scan_interval, uint16 scan_window, uint8 active);

/* Response id */
dumo_rsp_le_gap_set_scan_parameters_id

/* Response structure */
struct dumo_msg_le_gap_set_scan_parameters_rsp_t
{
    uint16 result
}
```

3.14.1.13 cmd_le_gap_set_scan_result_filter

This command can be used to filter scan responses and advertisements by device name.

Table 3.324. 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, Bluetooth Low Energy |
| 3 | 0x0a | method | Message ID |
| 4 | uint8array | name | Beginning of the device name in the payload of the scan responses and advertisements must match this string for the le_gap_scan_response to be generated. Each call to this function will add new filter in addition to previously set. To Clear the filtering issue le_gap_scan_filter_clear command By default and at boot filtering is disabled. |

Table 3.325. 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, Bluetooth Low Energy |
| 3 | 0x0a | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call le_gap_set_scan_result_filter(name_len, name_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_gap_set_scan_result_filter(uint8array name);

/* Response id */
dumo_rsp_le_gap_set_scan_result_filter_id

/* Response structure */
struct dumo_msg_le_gap_set_scan_result_filter_rsp_t
{
    uint16 result
}
```

3.14.2 le_gap events

3.14.2.1 evt_le_gap_host_channel_classification_complete

This event indicates LE host channel classification request is completed.

Table 3.326. Event

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | status | Result code <ul style="list-style-type: none">• 0: success• Non-zero: an error occurred For other values refer to the Error codes |

BGScript event

```
event le_gap_host_channel_classification_complete(status)
```

C Functions

```
/* Event id */
dumo_evt_le_gap_host_channel_classification_complete_id

/* Event structure */
struct dumo_msg_le_gap_host_channel_classification_complete_evt_t
{
    uint16 status
}
```

3.14.2.2 evt_le_gap_scan_response

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

Table 3.327. Event

| Byte | Type | Name | Description |
|------|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x0b | olen | Minimum payload length |
| 2 | 0x03 | class | Message class: Generic Access Profile, Bluetooth Low Energy |
| 3 | 0x00 | method | Message ID |
| 4 | int8 | rssi | Received signal strength indicator (RSSI) <ul style="list-style-type: none"> Range: -127 to +20 Units: dBm |
| 5 | uint8 | packet_type | Advertisement packet type <ul style="list-style-type: none"> 0x00: Connectable undirected advertising 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising 0x04: Scan Response Note: Scan response (0x04) is only received if the device is in active scan mode. |
| 6-11 | bd_addr | address | Bluetooth address of the remote device |
| 12 | uint8 | address_type | Advertiser address type. Values: <ul style="list-style-type: none"> 0: Public address 1: Random address |
| 13 | uint8 | bonding | Bonding handle if the remote advertising device has previously bonded with the local device. Values: <ul style="list-style-type: none"> 0xff: No bonding Other: Bonding handle |
| 14 | uint8array | data | Advertisement or scan response data |

BGScript event

```
event le_gap_scan_response(rssi,packet_type,address,address_type,bonding,data_len, data_data)
```

C Functions

```
/* Event id */
dumo_evt_le_gap_scan_response_id

/* Event structure */
struct dumo_msg_le_gap_scan_response_evt_t
{
    int8 rssi,
    uint8 packet_type,
    bd_addr *address,
    uint8 address_type,
    uint8 bonding,
    uint8array data
}
```

3.14.3 le_gap enumerations

3.14.3.1 enum_le_gap_address_type

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

Table 3.328. Enumerations

| Value | Name | Description |
|-------|-------------------------------------|----------------------------------------------|
| 0 | le_gap_address_type_public | LE public address |
| 1 | le_gap_address_type_random | LE random address |
| 2 | le_gap_address_type_public_identity | LE public identity address resolved by stack |
| 3 | le_gap_address_type_random_identity | LE random identity address resolved by stack |
| 16 | le_gap_address_type_bredr | Classic Bluetooth address |

3.14.3.2 enum_le_gap_connectable_mode

These values define the available Connectable Modes.

Table 3.329. Enumerations

| Value | Name | Description |
|-------|----------------------------------|--------------------------------------------------|
| 0 | le_gap_non_connectable | Not connectable |
| 1 | le_gap_directed_connectable | Directed Connectable (RESERVED, DO NOT USE) |
| 2 | le_gap_undirected_connectable | Undirected connectable |
| 3 | le_gap_scannable_non_connectable | Not connectable but responds to scan_req-packets |

3.14.3.3 enum_le_gap_discover_mode

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

Table 3.330. 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 |

3.14.3.4 enum_le_gap_discoverable_mode

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

Table 3.331. Enumerations

| Value | Name | Description |
|-------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | le_gap_non_discoverable | Not discoverable |
| 1 | le_gap_limited_discoverable | Discoverable using both limited and general discoverable mode |
| 2 | le_gap_general_discoverable | Discoverable using general discoverable mode |
| 3 | le_gap_broadcast | Device is not discoverable in either limited or generic discoverable procedure, but may be discovered by using the Observation procedure |
| 4 | le_gap_user_data | Send advertisement and/or scan response data defined by the user using le_gap_set_adv_data command . The limited/general discoverable flags are defined by the user. |

3.15 LE Cable Replacement (le_serial)

These commands and events are related to the establishment of Bluetooth LE cable replacement connections. LE cable replacement can be used to send and receive streaming data over LE connections. Please note that LE Cable Replacement commands are available only in images compiled with LE Cable Replacement sdk

3.15.1 le_serial commands

3.15.1.1 cmd_le_serial_listen

Listen for an incoming LE cable replacement connection. Creates an endpoint which will be set as active when an incoming connection completes. Note that once the cable replacement connection is closed by the remote side, the endpoint is not usable anymore; it must be cleaned up using [endpoint_close](#) command and listen must be called again to handle the next incoming connection. Also note that a listening endpoint may be closed only when not yet connected or already disconnected by the remote. Only the side initiating the connection may close the endpoint while it is connected.

Table 3.332. Command

| Byte | Type | Name | Description |
|------|--------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: LE Cable Replacement |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | characteristic | Handle of the local characteristic used to transmit data |
| 6 | uint8 | streaming_destination | Streaming destination endpoint for the connection. This should be one of the fixed endpoints. If this argument is set to the invalid endpoint handle (255), endpoint streaming will not be set up. Instead, bgscript can use endpoint_send command and endpoint_data event for sending and receiving data. |

Table 3.333. Response

| Byte | Type | Name | Description |
|------|--------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x14 | class | Message class: LE Cable Replacement |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | Endpoint ID assigned to the LE serial connection. Valid only if the result code is zero. Endpoint will become active when connection is ready. |

BGScript command

```
call le_serial_listen(characteristic,streaming_destination)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_serial_listen(uint16 characteristic, uint8 streaming_destination);

/* Response id */
dumo_rsp_le_serial_listen_id

/* Response structure */
struct dumo_msg_le_serial_listen_rsp_t
{
    uint16 result,
```

```
    uint8 endpoint  
}
```

Table 3.334. Events Generated

| Event | Description |
|----------------------------------|-------------------------------------------------------------------------------|
| le_serial_opened | This event indicates the establishment of an LE cable replacement connection. |

3.15.1.2 cmd_le_serial_open

Open an LE cable replacement connection endpoint to a remote device. The endpoint can be closed with the [endpoint_close](#) command when done. If no ACL connection to the remote device exists, one is formed before the cable replacement connection handshake. Otherwise the existing ACL connection will be used. Note that if the remote device requires encryption or authentication for communication, ACL connection security must be set up before opening the LE cable replacement connection; see [sm_increase_security](#) command

Table 3.335. Command

| Byte | Type | Name | Description |
|------|------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x09 | olen | Minimum payload length |
| 2 | 0x14 | class | Message class: LE Cable Replacement |
| 3 | 0x00 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of the remote device in little endian format. |
| 10 | uint8 | address_type | Address type of the device to connect to |
| 11 | uint8 | streaming_destination | Streaming destination for the connection endpoint. If this argument is set to the invalid endpoint handle (255), endpoint streaming will not be set up. Instead, bgscript can use endpoint_send command and endpoint_data event for sending and receiving data. |
| 12 | uint8array | service | UUID of the service to connect to. |

Table 3.336. Response

| Byte | Type | Name | Description |
|------|--------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x14 | class | Message class: LE Cable Replacement |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | endpoint | Endpoint ID assigned to the LE serial connection. Valid only if the result code is zero. Endpoint will become active when connection is ready. |

BGScript command

```
call le_serial_open(address,address_type,streaming_destination,service_len, service_data)(result,endpoint)
```

BGLIB C API

```
/* Function */
void dumo_cmd_le_serial_open(bd_addr *address, uint8 address_type, uint8 streaming_destination, uint8array service);

/* Response id */
dumo_rsp_le_serial_open_id
```

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

Table 3.337. Events Generated

| Event | Description |
|------------------|-------------------------------------------------------------------------------|
| le_serial_opened | This event indicates the establishment of an LE cable replacement connection. |

3.15.2 le_serial events

3.15.2.1 evt_le_serial_opened

This event indicates the establishment of an LE cable replacement connection.

Table 3.338. Event

| Byte | Type | Name | Description |
|------|-------|----------|----------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x14 | class | Message class: LE Cable Replacement |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | endpoint | Endpoint ID assigned to the connection |

BGScript event

```
event le_serial_opened(endpoint)
```

C Functions

```
/* Event id */
dumo_evt_le_serial_opened_id

/* Event structure */
struct dumo_msg_le_serial_opened_evt_t
{
    uint8 endpoint
}
```

3.16 Security Manager (sm)

The commands and events in this class are used to manage Bluetooth security and include commands for starting and stopping encryption and commands for managing bonding operations.

3.16.1 sm commands

3.16.1.1 cmd_sm_configure

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

Table 3.339. 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 | mitm_required | Require MITM flag. Values: • 0: Allow bonding without MITM protection • 1: Require MITM protection |
| 5 | uint8 | io_capabilities | I/O Capabilities. See link |

Table 3.340. 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 |

BGScript command

```
call sm_configure(mitm_required,io_capabilities)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_configure(uint8 mitm_required, uint8 io_capabilities);

/* Response id */
dumo_rsp_sm_configure_id

/* Response structure */
struct dumo_msg_sm_configure_rsp_t
{
    uint16 result
}
```

3.16.1.2 cmd_sm_delete_bonding

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

Table 3.341. 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 3.342. 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 |

BGScript command

```
call sm_delete_bonding(bonding)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_delete_bonding(uint8 bonding);

/* Response id */
dumo_rsp_sm_delete_bonding_id

/* Response structure */
struct dumo_msg_sm_delete_bonding_rsp_t
{
    uint16 result
}
```

3.16.1.3 cmd_sm_delete_bondings

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

Table 3.343. 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 3.344. 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 |

BGScript command

```
call sm_delete_bondings()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_delete_bondings();

/* Response id */
dumo_rsp_sm_delete_bondings_id

/* Response structure */
struct dumo_msg_sm_delete_bondings_rsp_t
{
    uint16 result
}
```

3.16.1.4 cmd_sm_enter_passkey

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

Table 3.345. Command

| Byte | Type | Name | Description |
|------|--------|------------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x08 | method | Message ID |
| 4 | uint8 | connection | Connection handle |
| 5-8 | uint32 | passkey | Passkey. Valid range: 0-999999 |

Table 3.346. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call sm_enter_passkey(connection,passkey)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_enter_passkey(uint8 connection, uint32 passkey);

/* Response id */
dumo_rsp_sm_enter_passkey_id

/* Response structure */
struct dumo_msg_sm_enter_passkey_rsp_t
{
    uint16 result
}
```

3.16.1.5 cmd_sm_enter_pin_code

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

Table 3.347. Command

| Byte | Type | Name | Description |
|------|------------|----------|-----------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x07 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0d | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of remote device |
| 10 | uint8array | pin_code | PIN code as a string. Length: 1-16 characters |

Table 3.348. 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 | 0x0d | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call sm_enter_pin_code(address,pin_code_len, pin_code_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_enter_pin_code(bd_addr *address, uint8array pin_code);

/* Response id */
dumo_rsp_sm_enter_pin_code_id

/* Response structure */
struct dumo_msg_sm_enter_pin_code_rsp_t
{
    uint16 result
}
```

3.16.1.6 cmd_sm_increase_security

This command can be used to enhance the security of a Bluetooth Classic or Bluetooth LE connection to current security requirements. By default, connections opened by the module are not encrypted, unless the remote device requests it (most devices do). For Bluetooth LE connections, this command initiates encryption, as there are only two levels of security: unencrypted and encrypted. If the devices are bonded, the existing bonding will be used. If the devices are not bonded, a new key will be created; if bonding is enabled the key will be stored for future use, if not the key will be discarded after the connection is closed. For Bluetooth Classic connections, in some cases it's possible to first increase the security to encryption with a Just Works key, then increase it again by creating a MITM protected key.

Table 3.349. Command

| Byte | Type | Name | Description |
|------|-------|------------|---------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x04 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

Table 3.350. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call sm_increase_security(connection)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_increase_security(uint8 connection);

/* Response id */
dumo_rsp_sm_increase_security_id

/* Response structure */
struct dumo_msg_sm_increase_security_rsp_t
{
    uint16 result
}
```

3.16.1.7 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 3.351. 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 3.352. 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 |

BGScript command

```
call sm_list_all_bondings()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_list_all_bondings();

/* Response id */
dumo_rsp_sm_list_all_bondings_id

/* Response structure */
struct dumo_msg_sm_list_all_bondings_rsp_t
{
    uint16 result
}
```

Table 3.353. 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. |

3.16.1.8 cmd_sm_passkey_confirm

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

Table 3.354. 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 3.355. 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 |

BGScript command

```
call sm_passkey_confirm(connection,confirm)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_passkey_confirm(uint8 connection, uint8 confirm);

/* Response id */
dumo_rsp_sm_passkey_confirm_id

/* Response structure */
struct dumo_msg_sm_passkey_confirm_rsp_t
{
    uint16 result
}
```

3.16.1.9 cmd_sm_read_bonding

This command can be used to read the encryption key for a specific bonding. Used in debugging for reading encryption keys which can be used e.g. for protocol sniffing.

Table 3.356. Command

| Byte | Type | Name | Description |
|------|-------|---------|---------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x05 | method | Message ID |
| 4 | uint8 | bonding | Bonding index of bonding data |

Table 3.357. Response

| Byte | Type | Name | Description |
|------|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x0a | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x05 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6-11 | bd_addr | address | Bluetooth address of the remote device this bonding entry refers to |
| 12 | uint8 | address_type | Address type of the this bonding entry |
| 13 | uint8array | bonding_key | Encryption key stored for this bonding entry. Maximum 16 bytes |

BGScript command

```
call sm_read_bonding(bonding)(result,address,address_type,bonding_key_len, bonding_key_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_read_bonding(uint8 bonding);

/* Response id */
dumo_rsp_sm_read_bonding_id

/* Response structure */
struct dumo_msg_sm_read_bonding_rsp_t
{
    uint16 result,
    bd_addr *address,
    uint8 address_type,
    uint8array bonding_key
}
```

3.16.1.10 cmd_sm_read_bonding_configuration

This command can be used to read the maximum number of allowed bonding entries and to reveal the currently set bonding policy.

Table 3.358. 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 | 0x03 | method | Message ID |

Table 3.359. Response

| Byte | Type | Name | Description |
|------|--------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | max_bonding_count | Maximum allowed bonding count. Range: 1 to 12 Default: 12 |
| 5 | uint8 | policy_flags | Bonding policy. Values: <ul style="list-style-type: none"> • 0: If database is full, new bonding attempts will fail • 1: New bonding will overwrite the oldest existing bonding |
| 6-7 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call sm_read_bonding_configuration()(max_bonding_count,policy_flags,result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_read_bonding_configuration();

/* Response id */
dumo_rsp_sm_read_bonding_configuration_id

/* Response structure */
struct dumo_msg_sm_read_bonding_configuration_rsp_t
{
    uint8 max_bonding_count,
    uint8 policy_flags,
    uint16 result
}
```

3.16.1.11 cmd_sm_set_bondable_mode

This command can be used to set whether the device accepts new bondings or not.

Table 3.360. 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 |

Table 3.361. 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 |

BGScript command

```
call sm_set_bondable_mode(bondable)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_set_bondable_mode(uint8 bondable);

/* Response id */
dumo_rsp_sm_set_bondable_mode_id

/* Response structure */
struct dumo_msg_sm_set_bondable_mode_rsp_t
{
    uint16 result
}
```

3.16.1.12 cmd_sm_set_minimum_key_size

This command can be used to set the minimum allowed encryption key size value used for bonding. The default value is 7 bytes. If negotiated value of the key size is less than expected minimum after connection establishment, the credential bonding process is considered invalid and [dumo_evt_sm_bonding_failed](#) event with [dumo_err_sm_encryption_key_size \(0x0306\)](#) code is generated.

Table 3.362. Command

| Byte | Type | Name | Description |
|------|-------|------------------|-----------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x10 | method | Message ID |
| 4 | uint8 | minimum_key_size | Minimum allowed encryption key size value for bonding. Range: 7 - 16 bytes. |

Table 3.363. Response

| Byte | Type | Name | Description |
|------|--------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x03 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x10 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |
| 6 | uint8 | minimum_key_size | Current setting of minimal allowed encryption key size value (in bytes). |

BGScript command

```
call sm_set_minimum_key_size(minimum_key_size)(result,minimum_key_size)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_set_minimum_key_size(uint8 minimum_key_size);

/* Response id */
dumo_rsp_sm_set_minimum_key_size_id

/* Response structure */
struct dumo_msg_sm_set_minimum_key_size_rsp_t
{
    uint16 result,
    uint8 minimum_key_size
}
```

3.16.1.13 cmd_sm_set_oob_data

This command can be used to set the OOB data (out-of-band encryption data) 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. This bonding method is available only for LE bonding.

Table 3.364. Command

| Byte | Type | Name | Description |
|------|------------|----------|--------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x0a | method | Message ID |
| 4 | uint8array | oob_data | LE OOB data. To set OOB data, send a 16-byte array. To clear OOB data, send a zero-length array. |

Table 3.365. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | 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 |

BGScript command

```
call sm_set_oob_data(oob_data_len, oob_data_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_set_oob_data(uint8array oob_data);

/* Response id */
dumo_rsp_sm_set_oob_data_id

/* Response structure */
struct dumo_msg_sm_set_oob_data_rsp_t
{
    uint16 result
}
```

3.16.1.14 cmd_sm_store_bonding_configuration

This command can be used to set maximum allowed bonding count and bonding policy.

Table 3.366. Command

| Byte | Type | Name | Description |
|------|-------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x02 | olen | 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 12 Default: 12 |
| 5 | uint8 | policy_flags | Bonding policy. Values: <ul style="list-style-type: none"> • 0: If database is full, new bonding attempts will fail • 1: New bonding will overwrite the oldest existing bonding Default: 0 |

Table 3.367. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call sm_store_bonding_configuration(max_bonding_count,policy_flags)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_sm_store_bonding_configuration(uint8 max_bonding_count, uint8 policy_flags);

/* Response id */
dumo_rsp_sm_store_bonding_configuration_id

/* Response structure */
struct dumo_msg_sm_store_bonding_configuration_rsp_t
{
    uint16 result
}
```

3.16.2 sm events

3.16.2.1 evt_sm_bonded

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

Table 3.368. Event

| Byte | Type | Name | Description |
|------|-------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | 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: <ul style="list-style-type: none">• 0xff: Pairing completed without bonding - the pairing key will be discarded after disconnection.• Other: Procedure completed, pairing key stored with given bonding handle |

BGScript event

```
event sm_bonded(connection,bonding)
```

C Functions

```
/* Event id */
dumo_evt_sm_bonded_id

/* Event structure */
struct dumo_msg_sm_bonded_evt_t
{
    uint8 connection,
    uint8 bonding
}
```

3.16.2.2 evt_sm_bonding_failed

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

Table 3.369. 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 |

BGScript event

```
event sm_bonding_failed(connection,reason)
```

C Functions

```
/* Event id */
dumo_evt_sm_bonding_failed_id

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

3.16.2.3 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 3.370. Event

| Byte | Type | Name | Description |
|------|--------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x05 | olen | 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. <ul style="list-style-type: none">• 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 |

BGScript event

```
event sm_confirm_passkey(connection,passkey)
```

C Functions

```
/* Event id */
dumo_evt_sm_confirm_passkey_id

/* Event structure */
struct dumo_msg_sm_confirm_passkey_evt_t
{
    uint8 connection,
    uint32 passkey
}
```

3.16.2.4 evt_sm_list_all_bondings_complete

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

Table 3.371. 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 |

BGScript event

```
event sm_list_all_bondings_complete()
```

C Functions

```
/* Event id */
dumo_evt_sm_list_all_bondings_complete_id

/* Event structure */
struct dumo_msg_sm_list_all_bondings_complete_evt_t
{
}
```

3.16.2.5 evt_sm_list_bonding_entry

This event is triggered by the command [sm_list_all_bondings](#) if bondings exist in the local database.

Table 3.372. Event

| Byte | Type | Name | Description |
|------|---------|--------------|----------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x08 | olen | 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 |

BGScript event

```
event sm_list_bonding_entry(bonding,address,address_type)
```

C Functions

```
/* Event id */
dumo_evt_sm_list_bonding_entry_id

/* Event structure */
struct dumo_msg_sm_list_bonding_entry_evt_t
{
    uint8 bonding,
    bd_addr *address,
    uint8 address_type
}
```

3.16.2.6 evt_sm_passkey_display

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

Table 3.373. Event

| Byte | Type | Name | Description |
|------|--------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x05 | olen | 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. <ul style="list-style-type: none">• 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 |

BGScript event

```
event sm_passkey_display(connection,passkey)
```

C Functions

```
/* Event id */
dumo_evt_sm_passkey_display_id

/* Event structure */
struct dumo_msg_sm_passkey_display_evt_t
{
    uint8 connection,
    uint32 passkey
}
```

3.16.2.7 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 3.374. Event

| Byte | Type | Name | Description |
|------|-------|------------|---------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | connection | Connection handle |

BGScript event

```
event sm_passkey_request(connection)
```

C Functions

```
/* Event id */
dumo_evt_sm_passkey_request_id

/* Event structure */
struct dumo_msg_sm_passkey_request_evt_t
{
    uint8 connection
}
```

3.16.2.8 evt_sm_pin_code_request

This event indicates a request for the user to enter the PIN. Use the command [sm_enter_pin_code](#) to input the PIN code value. The PIN code is used to create legacy pairing with old (v2.0 and before) Bluetooth devices.

Table 3.375. Event

| Byte | Type | Name | Description |
|------|---------|---------|------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x06 | olen | Minimum payload length |
| 2 | 0x0f | class | Message class: Security Manager |
| 3 | 0x08 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth address of remote device |

BGScript event

```
event sm_pin_code_request(address)
```

C Functions

```
/* Event id */
dumo_evt_sm_pin_code_request_id

/* Event structure */
struct dumo_msg_sm_pin_code_request_evt_t
{
    bd_addr *address
}
```

3.16.3 sm enumerations

3.16.3.1 enum_sm_io_capability

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

Table 3.376. 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 |

3.17 System (system)

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

3.17.1 system commands

3.17.1.1 cmd_system_get_bt_address

This command can be used to read the Bluetooth address of the module.

Table 3.377. 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 3.378. 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 Classic address and LE public address in little endian format. |

BGScript command

```
call system_get_bt_address() (address)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_get_bt_address();

/* Response id */
dumo_rsp_system_get_bt_address_id

/* Response structure */
struct dumo_msg_system_get_bt_address_rsp_t
{
    bd_addr *address
}
```

3.17.1.2 cmd_system_get_class_of_device

This command can be used to read the device's Bluetooth Classic Class of Device (COD) information.

Table 3.379. Command

| Byte | Type | Name | Description |
|------|------|--------|------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x05 | method | Message ID |

Table 3.380. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x06 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x05 | method | Message ID |
| 4-7 | uint32 | cod | Class of Device. A comprehensive list of values can be found at https://www.bluetooth.org/en-us/specification/assigned-numbers/baseband <ul style="list-style-type: none"> Example: 0x001f00: Uncategorized device |
| 8-9 | uint16 | result | Result code <ul style="list-style-type: none"> 0: success Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call system_get_class_of_device()(cod,result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_get_class_of_device();

/* Response id */
dumo_rsp_system_get_class_of_device_id

/* Response structure */
struct dumo_msg_system_get_class_of_device_rsp_t
{
    uint32 cod,
    uint16 result
}
```

3.17.1.3 cmd_system_get_local_name

This command can be used to read the Bluetooth Classic friendly name of the local device. Note that for Bluetooth LE the device name is stored in the GAP Service of the GATT database.

Table 3.381. Command

| Byte | Type | Name | Description |
|------|------|--------|------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x09 | method | Message ID |

Table 3.382. Response

| Byte | Type | Name | Description |
|------|------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 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 | name | Local device's Bluetooth Classic friendly name |

BGScript command

```
call system_get_local_name()(result,name_len, name_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_get_local_name();

/* Response id */
dumo_rsp_system_get_local_name_id

/* Response structure */
struct dumo_msg_system_get_local_name_rsp_t
{
    uint16 result,
    uint8array name
}
```

3.17.1.4 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 module is working.

Table 3.383. 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 3.384. 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 |

BGScript command

```
call system_hello()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_hello();

/* Response id */
dumo_rsp_system_hello_id

/* Response structure */
struct dumo_msg_system_hello_rsp_t
{
    uint16 result
}
```

3.17.1.5 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 3.385. Command

| Byte | Type | Name | Description |
|------|-------|--------|--------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | 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: <ul style="list-style-type: none">• 0: Normal reset• 1: Boot to DFU mode |

BGScript command

```
call system_reset(dfu)
```

BGLIB C API

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

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

Table 3.386. Events Generated

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

3.17.1.6 cmd_system_reset_factory_settings

This command can be used to reset the module and clear all settings including bonding information. It also resets the friendly name to its default value. The Apple iAP configuration and the persistent store are cleared. Note that the module's Bluetooth Classic / LE public address is preserved.

Table 3.387. Command

| Byte | Type | Name | Description |
|------|------|--------|------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x07 | method | Message ID |

Table 3.388. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 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 |

BGScript command

```
call system_reset_factory_settings()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_reset_factory_settings();

/* Response id */
dumo_rsp_system_reset_factory_settings_id

/* Response structure */
struct dumo_msg_system_reset_factory_settings_rsp_t
{
    uint16 result
}
```

3.17.1.7 cmd_system_set_class_of_device

This command is used to set the Bluetooth Classic Class of Device (COD) setting.

Table 3.389. Command

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x06 | method | Message ID |
| 4-7 | uint32 | cod | Class of Device. A comprehensive list of values can be found at https://www.bluetooth.org/en-us/specification/assigned-numbers/baseband <ul style="list-style-type: none"> Example: 0x001f00: Uncategorized device |

Table 3.390. Response

| Byte | Type | Name | Description |
|------|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x06 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> 0: success Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call system_set_class_of_device(cod)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_set_class_of_device(uint32 cod);

/* Response id */
dumo_rsp_system_set_class_of_device_id

/* Response structure */
struct dumo_msg_system_set_class_of_device_rsp_t
{
    uint16 result
}
```

3.17.1.8 cmd_system_set_local_name

This command can be used to set the local device's Bluetooth Classic friendly name. For Bluetooth LE the device name is stored in the GAP Service local name parameter of the GATT database. If the device name attribute is not set as a constant in the project's GATT configuration XML file, it can be changed with [gatt_server_write_attribute_value](#). It is also possible to advertise with a different device name by using [le_gap_set_adv_data](#), but in that case if a remote device connects and asks for the device name over GATT, the value will be different from the advertised value.

Table 3.391. Command

| Byte | Type | Name | Description |
|------|------------|--------|---------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x08 | method | Message ID |
| 4 | uint8array | name | Local device's Bluetooth Classic friendly name. Maximum name length is 30 bytes |

Table 3.392. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 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 |

BGScript command

```
call system_set_local_name(name_len, name_data)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_set_local_name(uint8array name);

/* Response id */
dumo_rsp_system_set_local_name_id

/* Response structure */
struct dumo_msg_system_set_local_name_rsp_t
{
    uint16 result
}
```

3.17.1.9 cmd_system_set_max_power_mode

This command can be used to set the most efficient power saving state allowed for the system.

Table 3.393. Command

| Byte | Type | Name | Description |
|------|-------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x02 | method | Message ID |
| 4 | uint8 | power_mode | Power saving mode. Values: <ul style="list-style-type: none"> • 1: CPU allowed to enter idle mode • 2: CPU allowed to enter idle and sleep modes Default: 2. See the module data sheet for details concerning power modes. Power saving modes 1 and 2 differ only if sleep has been enabled in the HW configuration. This setting is not persistent |

Table 3.394. Response

| Byte | Type | Name | Description |
|------|--------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x02 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call system_set_max_power_mode(power_mode)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_system_set_max_power_mode(uint8 power_mode);

/* Response id */
dumo_rsp_system_set_max_power_mode_id

/* Response structure */
struct dumo_msg_system_set_max_power_mode_rsp_t
{
    uint16 result
}
```

3.17.2 system events

3.17.2.1 evt_system_boot

This event indicates the device has started and is ready to receive commands that are not related to Bluetooth. When the Bluetooth stack is ready, the event `system_initialized` is generated. This event carries the firmware build number and other SW and HW identification codes.

Table 3.395. Event

| Byte | Type | Name | Description |
|-------|--------|------------|------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x0c | 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 |
| 12-13 | uint16 | bootloader | Bootloader version |
| 14-15 | uint16 | hw | Hardware type |

BGScript event

```
event system_boot(major,minor,patch,build,bootloader,hw)
```

C Functions

```
/* Event id */
dumo_evt_system_boot_id

/* Event structure */
struct dumo_msg_system_boot_evt_t
{
    uint16 major,
    uint16 minor,
    uint16 patch,
    uint16 build,
    uint16 bootloader,
    uint16 hw
}
```

3.17.2.2 evt_system_initialized

This event indicates that all systems including the Bluetooth stack and radio are ready for use.

Table 3.396. Event

| Byte | Type | Name | Description |
|------|---------|---------|--------------------------------------------------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x06 | olen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x01 | method | Message ID |
| 4-9 | bd_addr | address | Bluetooth Classic address and LE public address in little endian format. |

BGScript event

```
event system_initialized(address)
```

C Functions

```
/* Event id */
dumo_evt_system_initialized_id

/* Event structure */
struct dumo_msg_system_initialized_evt_t
{
    bd_addr *address
}
```

3.17.2.3 evt_system_recovery

This event indicates the device has encountered an error condition and has reset.

Table 3.397. Event

| Byte | Type | Name | Description |
|------|------------|--------|------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x09 | olen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x02 | method | Message ID |
| 4-7 | uint32 | id1 | |
| 8-11 | uint32 | id2 | |
| 12 | uint8array | data | |

BGScript event

```
event system_recovery(id1,id2,data_len, data_data)
```

C Functions

```
/* Event id */
dumo_evt_system_recovery_id

/* Event structure */
struct dumo_msg_system_recovery_evt_t
{
    uint32 id1,
    uint32 id2,
    uint8array data
}
```

3.17.2.4 evt_system_script_stopped

This event is generated when a event handler has been running more than 1 000 000 script intepreter steps (about 2 seconds).

Table 3.398. Event

| Byte | Type | Name | Description |
|------|-------|-------------|---------------------------------|
| 0 | 0xa0 | hlen | Message type: Event |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0x01 | class | Message class: System |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | event_class | Id to to Class of handled event |
| 5 | uint8 | event_id | event id of handled event |

BGScript event

```
event system_script_stopped(event_class,event_id)
```

C Functions

```
/* Event id */
dumo_evt_system_script_stopped_id

/* Event structure */
struct dumo_msg_system_script_stopped_evt_t
{
    uint8 event_class,
    uint8 event_id
}
```

3.18 Testing commands (test)

The commands and events in this class can be used in production and RF development testing and as an aid in debugging.

3.18.1 test commands

3.18.1.1 cmd_test_device_under_test_mode

This command can be used to set BT121 in "BT RF SIG mode" which is meant for connecting with a Bluetooth tester where the BT121 is controlled over the LMP (Link Management Protocol). Command makes the BT121 module visible, connectable, acceptable all connections and put them in test mode. Then the Bluetooth tester can take control of the BT121 module through the RF link. For more details see Bluetooth Core Specification 4.1 (Volume 2 Part E, chapter 7.6.3).

Table 3.399. Command

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0xe0 | class | Message class: Testing commands |
| 3 | 0x08 | method | Message ID |

Table 3.400. Response

| Byte | Type | Name | Description |
|------|--------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | lolen | Minimum payload length |
| 2 | 0xe0 | class | Message class: Testing commands |
| 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 |

BGScript command

```
call test_device_under_test_mode()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_device_under_test_mode();

/* Response id */
dumo_rsp_test_device_under_test_mode_id

/* Response structure */
struct dumo_msg_test_device_under_test_mode_rsp_t
{
    uint16 result
}
```

3.18.1.2 cmd_test_dtm_end

This command can be used to end the Bluetooth LE Direct Test Mode (DTM) mode.

Table 3.401. 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 3.402. 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 |

BGScript command

```
call test_dtm_end()(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_dtm_end();

/* Response id */
dumo_rsp_test_dtm_end_id

/* Response structure */
struct dumo_msg_test_dtm_end_rsp_t
{
    uint16 result
}
```

Table 3.403. Events Generated

| Event | Description |
|--------------------|-----------------------------------------------|
| test_dtm_completed | LE Direct Test Mode TX (or RX) has completed. |

3.18.1.3 cmd_test_dtm_rx

This command can be used to start the Bluetooth LE Direct Test Mode (DTM) RX test. After RX test termination with [dumo_cmd_test_dtm_end](#), event [dumo_evt_test_dtm_completed](#) (with number of packets received) will be generated.

Table 3.404. Command

| Byte | Type | Name | Description |
|------|-------|---------|--------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x01 | method | Message ID |
| 4 | uint8 | channel | Bluetooth channel to use in testing. Value range: 0-39 |

Table 3.405. Response

| Byte | Type | Name | Description |
|------|--------|--------|---------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x01 | method | Message ID |
| 4-5 | uint16 | result | Command result |

BGScript command

```
call test_dtm_rx(channel)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_dtm_rx(uint8 channel);

/* Response id */
dumo_rsp_test_dtm_rx_id

/* Response structure */
struct dumo_msg_test_dtm_rx_rsp_t
{
    uint16 result
}
```

3.18.1.4 cmd_test_dtm_tx

This command can be used to start the Bluetooth LE Direct Test Mode TX test. The module will send continuous BLE packets at the highest output power for BLE. It is possible to change the RF TX power in DTM transmissions using [dumo_cmd_le_gap_set_max_power](#) command before starting the test mode.

Table 3.406. Command

| Byte | Type | Name | Description |
|------|--------|-------------------|--------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x05 | olen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x00 | method | Message ID |
| 4 | uint8 | packet_type | Packet type to use in testing. Value 3 corresponds to unmodulated carrier. Value range: 0-3 |
| 5 | uint8 | length | Packet length to use in testing. Value range: 0-37 |
| 6 | uint8 | channel | Bluetooth channel to use in testing. Value range: 0-39 |
| 7-8 | uint16 | number_of_packets | Number of packets to transmit. Value range: 0-65536. If 0 chosen as a value the continuous packet transmission selected. |

Table 3.407. Response

| Byte | Type | Name | Description |
|------|--------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x02 | olen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x00 | method | Message ID |
| 4-5 | uint16 | result | Command result |

BGScript command

```
call test_dtm_tx(packet_type,length,channel,number_of_packets)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_dtm_tx(uint8 packet_type, uint8 length, uint8 channel, uint16 number_of_packets);

/* Response id */
dumo_rsp_test_dtm_tx_id

/* Response structure */
struct dumo_msg_test_dtm_tx_rsp_t
{
    uint16 result
}
```

3.18.1.5 cmd_test_packet_test

This command can be used to start the packet testing mode for Bluetooth Classic.

Table 3.408. Command

| Byte | Type | Name | Description |
|------|--------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x08 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x0a | method | Message ID |
| 4 | uint8 | mode | Values: • 0: Frequency hopping enabled • 3: Single frequency |
| 5 | uint8 | tx_freq | Bluetooth TX channel/frequency to use in single frequency mode. Value range is 0-78, where • $F = 2402 + 2k$, when k from 0 to 39 • $F = 2403 + 2(k-40)$, when k from 40 to 78 |
| 6 | uint8 | rx_freq | Bluetooth RX channel/frequency to use in single frequency mode. Value range is 0-78, where • $F = 2402 + 2k$, when k from 0 to 39 • $F = 2403 + 2(k-40)$, when k from 40 to 78 • Disable RX (packet TX only), when k = 255 |
| 7 | uint8 | acl_type | See http://processors.wiki.ti.com/index.php/CC256x_VS_HCI_Commands#HCI_VS_DRPb_Tester_Packet_TX_RX_.280xFD85.29 for values and their meaning |
| 8-9 | uint16 | acl_len | See http://processors.wiki.ti.com/index.php/CC256x_VS_HCI_Commands#HCI_VS_DRPb_Tester_Packet_TX_RX_.280xFD85.29 for values and their meaning |
| 10 | uint8 | power | TX power. Value range: 8-15, 8 is lowest, 15 is highest. See http://processors.wiki.ti.com/index.php/CC256x_VS_HCI_Commands#HCI_VS_DRPb_Set_Power_Vector_.280xFD82.29 for actual dBm values. |
| 11 | uint8 | disable_whitening | 0 to enable whitening, 1 to disable whitening |

Table 3.409. 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 | 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 |

BGScript command

```
call test_packet_test(mode,tx_freq,rx_freq,acl_type,acl_len,power,disable_whitening)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_packet_test(uint8 mode, uint8 tx_freq, uint8 rx_freq, uint8 acl_type, uint16 acl_len, uint8 power, uint8 disable_whitening);

/* Response id */
dumo_rsp_test_packet_test_id

/* Response structure */
struct dumo_msg_test_packet_test_rsp_t
{
    uint16 result
}
```

3.18.1.6 cmd_test_rx_test

This command can be used to start the RX test mode with continuous reception. It can be useful for current consumption measurements or marking the level of unwanted emissions.

Table 3.410. Command

| Byte | Type | Name | Description |
|------|-------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x09 | method | Message ID |
| 4 | uint8 | channel | Bluetooth RX channel/frequency to use. Value range is 0-78, where <ul style="list-style-type: none"> • F = 2402 + 2k, when k from 0 to 39 • F = 2403 + 2(k-40) when k from 40 to 78 |

Table 3.411. 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 | 0x09 | method | Message ID |
| 4-5 | uint16 | result | Result code <ul style="list-style-type: none"> • 0: success • Non-zero: an error occurred For other values refer to the Error codes |

BGScript command

```
call test_rx_test(channel)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_rx_test(uint8 channel);

/* Response id */
dumo_rsp_test_rx_test_id

/* Response structure */
struct dumo_msg_test_rx_test_rsp_t
{
    uint16 result
}
```

3.18.1.7 cmd_test_ssp_debug

This command can be used to enable or disable using a pre-defined Diffie-Hellman key pair for generating Bluetooth Classic Secure Simple Pairing link keys. When the debug mode is enabled, a Bluetooth sniffer can decrypt the communication between two Bluetooth devices using Secure Simple Pairing. It is sufficient that one party uses the debug mode - it is not necessary for both parties to use it.

Table 3.412. Command

| Byte | Type | Name | Description |
|------|-------|--------|-----------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x06 | method | Message ID |
| 4 | uint8 | enable | Debug enable flag. Values: • 0: Disable • 1: Enable |

Table 3.413. Response

| Byte | Type | Name | Description |
|------|------|--------|---------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x00 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x06 | method | Message ID |

BGScript command

```
call test_ssp_debug(enable)()
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_ssp_debug(uint8 enable);

/* Response id */
dumo_rsp_test_ssp_debug_id

/* Response structure */
struct dumo_msg_test_ssp_debug_rsp_t
{
}
```

3.18.1.8 cmd_test_tx_test

This command can be used to start a continuous transmission TX test. For Continuous Wave modulation, the data is all zeros, for all other modulations the data is PN15 pseudo - random noise. No Bluetooth - formatted packets are sent in this mode.

Table 3.414. Command

| Byte | Type | Name | Description |
|------|-------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x03 | lolen | Minimum payload length |
| 2 | 0x0e | class | Message class: Testing commands |
| 3 | 0x03 | method | Message ID |
| 4 | uint8 | modulation | Modulation type. Values: • 0 : Continuous Wave • 1 : 1 Mbit GFSK • 2 : 2 Mbit Pi/4-DQPSK • 3 : 3 Mbit 8-PSK • 4 : 1 Mbit GFSK (Bluetooth LE) |
| 5 | uint8 | channel | Bluetooth channel/frequency to use in testing. For modulation types 0-3, the value range is 0-78, where • $F = 2402 + 2k$, when k from 0 to 39 • $F = 2403 + 2(k-40)$ when k from 40 to 78 For modulation type 4, the range is 0-39. |
| 6 | uint8 | power | TX power level. Range: 1, 8-15, where: • 0 : Reserved • 1 : Bluetooth LE TX power (used when modulation is 4) • 2-7 : Reserved • 8-15 : Bluetooth Classic TX power, 8 is lowest, 15 is highest For the actual transmit power levels in dBm, see http://processors.wiki.ti.com/index.php/CC256x_VS_HCI_Commands#HCI_VS_DRPb_Set_Power_Vector_.280xFD82.29 |

Table 3.415. 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 | 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 |

BGScript command

```
call test_tx_test(modulation,channel,power)(result)
```

BGLIB C API

```
/* Function */
void dumo_cmd_test_tx_test(uint8 modulation, uint8 channel, uint8 power);

/* Response id */
dumo_rsp_test_tx_test_id

/* Response structure */
struct dumo_msg_test_tx_test_rsp_t
{
    uint16 result
}
```

3.18.2 test events**3.18.2.1 evt_test_dtm_completed**

This event indicates that a Bluetooth LE Direct Test Mode (DTM) command has completed.

Table 3.416. Event

| Byte | Type | Name | Description |
|------|--------|-------------------|---------------------------------------------------------------------------------|
| 0 | 0xa0 | hilen | Message type: Event |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0xe0 | 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 received, only valid for test_dtm_end command |

BGScript event

```
event test_dtm_completed(result,number_of_packets)
```

C Functions

```
/* Event id */
dumo_evt_test_dtm_completed_id

/* Event structure */
struct dumo_msg_test_dtm_completed_evt_t
{
    uint16 result,
    uint16 number_of_packets
}
```

3.18.3 test enumerations

3.18.3.1 enum_test_packet_type

The following table lists supported test packet types.

Table 3.417. 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 | Unmodulated carrier |

3.19 Utilities for BGScript (util)

The commands and events in this class can be used to simplify BGScript based application development and are typically not used in any other applications.

3.19.1 util commands

3.19.1.1 cmd_util_atoi

Converts decimal value in ASCII string to 32-bit signed integer.

Table 3.418. Command

| Byte | Type | Name | Description |
|------|------------|--------|---------------------------------------|
| 0 | 0x20 | hlen | Message type: Command |
| 1 | 0x01 | olen | Minimum payload length |
| 2 | 0x11 | class | Message class: Utilities for BGScript |
| 3 | 0x00 | method | Message ID |
| 4 | uint8array | string | String to convert |

Table 3.419. Response

| Byte | Type | Name | Description |
|------|-------|--------|---------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hlen | Message type: Response |
| 1 | 0x04 | olen | Minimum payload length |
| 2 | 0x11 | class | Message class: Utilities for BGScript |
| 3 | 0x00 | method | Message ID |
| 4-7 | int32 | value | Conversion result presenting the decimal value input as a string as a 32-bit signed integer value |

BGScript command

```
call util_atoi(string_len, string_data)(value)
```

BGLIB C API

```
/* Function */
void dumo_cmd_util_atoi(uint8array string);

/* Response id */
dumo_rsp_util_atoi_id

/* Response structure */
struct dumo_msg_util_atoi_rsp_t
{
    int32 value
}
```

3.19.1.2 cmd_util_itoa

This command can be used to convert a 32-bit signed integer value into a decimal value represented as a string.

Table 3.420. Command

| Byte | Type | Name | Description |
|------|-------|--------|---------------------------------------|
| 0 | 0x20 | hilen | Message type: Command |
| 1 | 0x04 | lolen | Minimum payload length |
| 2 | 0x11 | class | Message class: Utilities for BGScript |
| 3 | 0x01 | method | Message ID |
| 4-7 | int32 | value | 32-bit number to convert |

Table 3.421. Response

| Byte | Type | Name | Description |
|------|------------|--------|-------------------------------------------------------------------------------------------------------------|
| 0 | 0x20 | hilen | Message type: Response |
| 1 | 0x01 | lolen | Minimum payload length |
| 2 | 0x11 | class | Message class: Utilities for BGScript |
| 3 | 0x01 | method | Message ID |
| 4 | uint8array | string | Conversion result presenting the 32-bit signed integer value input as a decimal value presented by a string |

BGScript command

```
call util_itoa(value)(string_len, string_data)
```

BGLIB C API

```
/* Function */
void dumo_cmd_util_itoa(int32 value);

/* Response id */
dumo_rsp_util_itoa_id

/* Response structure */
struct dumo_msg_util_itoa_rsp_t
{
    uint8array string
}
```

3.20 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. |

| Code | Name | Description |
|--------|-------------------|-----------------------------------------------|
| 0x0504 | i2c_timeout | I2C read or write timed out. |
| 0x0505 | not_configured | Hardware is not configured for that function. |
| 0x0506 | ble_not_supported | Hardware does not support Bluetooth Smart. |

■ 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 |
| 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 | server_already_in_use | A server for the specified type of connection (SPP, HID, or L2CAP with specific PSM) was already started. Only one SPP server per SDP entry is supported; only one HID server altogether is supported; only one L2CAP sever per PSM is supported. |
| 0x0193 | no_such_endpoint | The specified endpoint does not exist. |
| 0x0194 | invalid_endpoint | The endpoint exists, but it does not support the BGAPI command executed. |

| Code | Name | Description |
|--------|------------------------|------------------------------------------------------------------|
| 0x0195 | invalid_endpoint_state | The endpoint could not execute the command in its current state. |

■ SDP errors

| Code | Name | Description |
|--------|----------------------|------------------------------------------------|
| 0x0601 | record_not_found | Service Record not found |
| 0x0602 | record_already_exist | Service Record with this handle already exist. |

■ 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 |
| 0x030a | invalid_parameters | The Invalid Parameters error code indicates: the command length is invalid or a parameter is outside of the specified range. |
| 0x030b | no_bonding | The bonding does not exist. |

■ Bluetooth errors

| Code | Name | Description |
|--------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0x0202 | unknown_connection_identifier | A command was sent from the Host that should identify a connection, but that connection does not exist. |
| 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 |

| Code | Name | Description |
|--------|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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_connection_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. |
| 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. |

| Code | Name | Description |
|--------|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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. |
| 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_slotViolation | 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. |

| Code | Name | Description |
|--------|------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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. |
| 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. |
| 0x0241 | refused_psm_not_supported | The remote end did not have a service that would accept L2CAP connection requests to the specified PSM. |
| 0x0242 | refused_security_block | The L2CAP connection attempt was blocked by the remote end due to security reasons. |
| 0x0243 | refused_no_resources | The remote end rejected the L2CAP connection due to lack of resources. |
| 0x0244 | acl_disconnected | The ACL carrying the L2CAP link was disconnected. |
| 0x0245 | psm_already_in_use | An L2CAP connection with the specified PSM is already open on the same ACL. For each ACL, only one connection can use the same PSM. |

■ 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. |

■ 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 queued |
| 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 |
| 0x0480 | application | Application error code defined by a higher layer specification. |

■ Filesystem errors

| Code | Name | Description |
|--------|----------------|----------------|
| 0x0901 | file_not_found | File not found |

4. Document Revision History

Table 4.1. Document Revision History

| Revision Number | Effective Date | Change Description |
|-----------------|-----------------|------------------------------------|
| 1.0 | 1 April 2015 | Initial version |
| 1.1 | 7 July 2016 | Updated for software version 1.1.0 |
| 1.2 | 1 November 2016 | Updated for software version 1.1.1 |
| 1.3 | 12 January 2017 | Updated for software version 1.1.2 |
| 1.4 | 28 August 2018 | Updated for software version 1.2.0 |
| 1.5 | 25 October 2019 | Updated for software version 1.3.0 |