# BLETOOL Manual

Version: BLETOOL \_Manual\_V0.7

Date: 2020-12-16

History

|  |  |  |
| --- | --- | --- |
| Version | Date | Description |
| 0.1 | 2019-04-18 | Initial |
| 0.2 | 2019-05-06 | Add connection, gatt operation commands |
| 0.3 | 2020-02-06 | Delete some commands and fix calibrate parameters and return values | 2019-5-06 | Add connection, gatt operation commands and error codes |
| 0.4 | 2020-03-26 | Add C/C++ API |  |  |
| 0.5 | 2020-03-30 | Change introductions |  |  |
| 0.6 | 2020-06-22 | Modify the callback function & the API parameters |  |  |
| 0.7 | 2020-12-15 | Change API parameters (" Connect "parameter changed to" Address "parameter) |  |  |

# Table of Contents

[BLETOOL Manual 1](#_Toc59549015)

[Table of Contents 2](#_Toc59549016)

[1. Description 4](#_Toc59549017)

[1.1 What’s bletool 4](#_Toc59549018)

[1.2 How to install 4](#_Toc59549019)

[1.3 How to use 5](#_Toc59549020)

[2. API References 6](#_Toc59549021)

[2.1 enable 6](#_Toc59549022)

[2.2 local\_address 6](#_Toc59549023)

[2.3 set\_power 7](#_Toc59549024)

[2.4 listen 8](#_Toc59549025)

[2.4.1 int32\_t (\*ble\_module\_event)(gl\_ble\_module\_event\_t event, gl\_ble\_module\_data\_t \*data); 8](#_Toc59549026)

[2.4.2 int32\_t (\*ble\_gap\_event)(gl\_ble\_gap\_evrnt\_t event, gl\_ble\_gap\_data\_t \*data); 9](#_Toc59549027)

[2.4.3 int32\_t (\*ble\_gatt\_event)(gl\_ble\_gatt\_event\_t event, gl\_ble\_gatt\_data\_t \*data); 13](#_Toc59549028)

[2.5 adv\_data 15](#_Toc59549029)

[2.6 adv 15](#_Toc59549030)

[2.7 adv\_stop 17](#_Toc59549031)

[2.8 send\_notify 18](#_Toc59549032)

[2.9 discovery 18](#_Toc59549033)

[2.10 stop 20](#_Toc59549034)

[2.11 connect 21](#_Toc59549035)

[2.12 disconnect 23](#_Toc59549036)

[2.13 get\_rssi 23](#_Toc59549037)

[2.14 get\_service 25](#_Toc59549038)

[2.15 get\_char 26](#_Toc59549039)

[2.16 set\_notify 28](#_Toc59549040)

[2.17 read\_value 29](#_Toc59549041)

[2.18 write\_value 30](#_Toc59549042)

# 1. Description

## 1.1 What’s bletool

**BleTool** is a software develop kit for Bluetooth Low Energy (BLE) in GL-iNET’s products. It provides a basic and simple method for developers to operate all the BLE functions.

Different from BlueZ which includes the full Bluetooth protocol stack in the host system, bletool is a light weight tool to operate hostless BLE modules which has fully built-in protocol stack. The module can fully operate on itself rather than depending on the host system.

To use BleTool, you need to have one of the following devices.

* GL-S1300 (Convexa-S): Smarthome gateway with beamforming Wi-Fi
* GL-S100: Smarthome gateway with 2.4G Wi-Fi
* GL-X750 (Spitz): LTE IoT gateway
* GL-B2200 (): Whole home mesh system and gateway

You can also use BleTool if you use Silconlabs EFR32 BLE modules which use UART/SPI to connect to your host Linux.

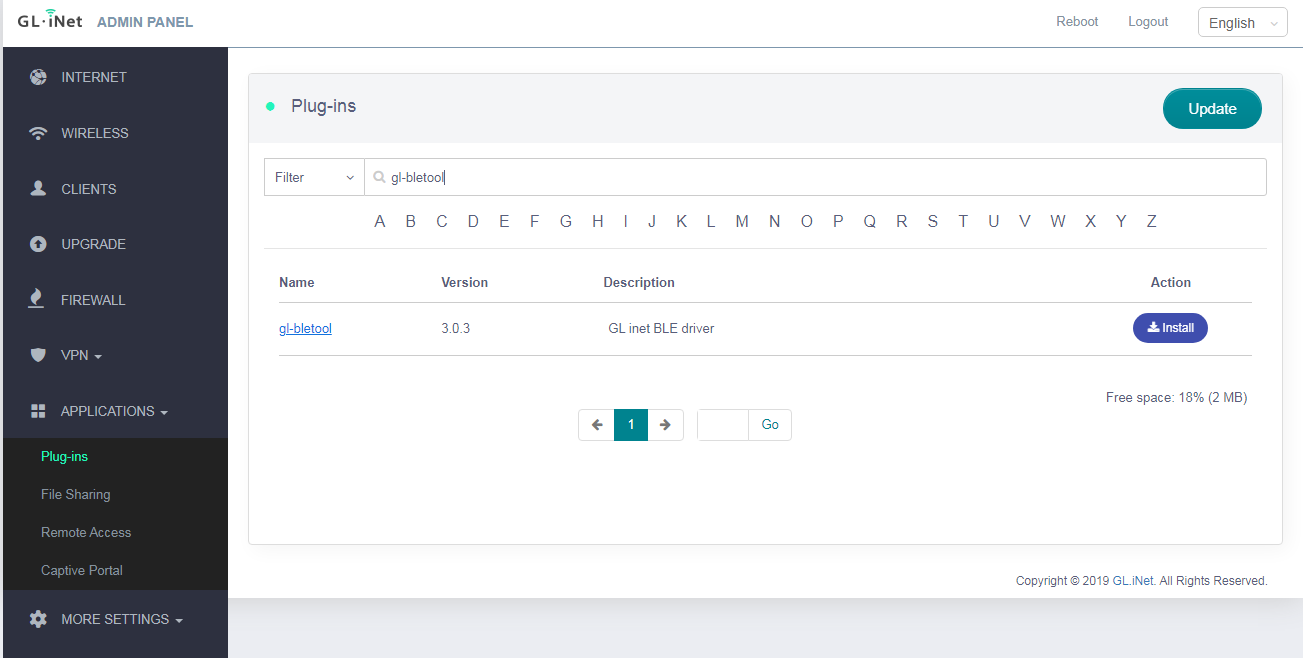
## 1.2 How to install

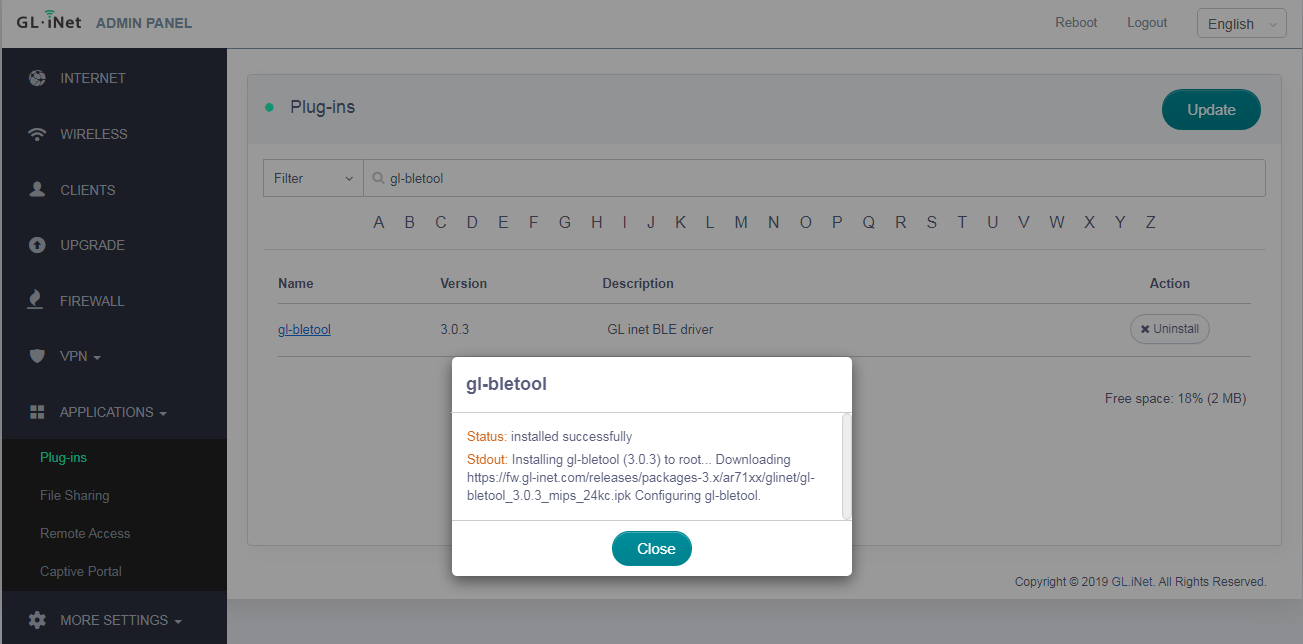
By default, BleTool is not installed on your router. You can install it using opkg if you can ssh to the router.

opkg update

opkg install gl-bletool

Alternatively, you can install using the web UI. Login your router’s web UI using your browser which is http://192.168.8.1 by default. Then go to APPLICATIONS->Plug-ins. First click “Update” to refresh your software repo then search “gl-bletool”. Click “install” and wait until you got “installation successfully”.





## 1.3 How to use

BleTool provides the following elements to handle BLE advertising, connection and GATT services.

* C/C++ APIs: This includes C functions, C header files based on which you can write your own code.
* C/C++ library: You can link this library with your own C application. You need to include the C header files in your own code to compile.
* cli (command line) tools: cli is commands that you can run in Linux terminal. You can use cli tools to test your BLE applications quickly and easily.

Here is example of how to use cli commands.



Below is the details of the API reference as well as the cli commands.

# 2. API References

Note that each API function will generate a message and pass to its fixed structure parameter after been called. It is a pointer to a structure. This should be appointed by user to handle the message.

## 2.1 enable

Enable or disable the BLE hardware.

**C API:**

int32\_t gl\_ble\_enable(int32\_t enable);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| int32\_t | enable | 0 means disable the BLE hardware;  None-zero means enable the BLE hardware. | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool enable 1

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value\* | Description |
| int32\_t | enable | 1 | 0 means disable the BLE hardware;  None-zero means enable the BLE hardware. |

Note that must call this command or API before using any other BLE commands or functions.

*\*A default value means you may not set this parameter. “-” means you must set this parameter.*

## 2.2 local\_address

Get the Local Bluetooth MAC address.

**C API:**

int32\_t gl\_ble\_get\_mac(gl\_ble\_get\_mac\_rsp\_t \*rsp);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure that stores local Bluetooth MAC address | |

#define DEVICE\_MAC\_LEN 6

typedef struct {

uint8\_t address[DEVICE\_MAC\_LEN];

} gl\_ble\_get\_mac\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_get\_mac\_rsp\_t | | |
| Type | Name | Description | |
| uint8\_t | address | The array of local Bluetooth MAC address | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| string | address | Local Bluetooth address like “11:22:33:44:55:66” | |

**CLI command:**

bletool local\_address

## 2.3 set\_power

Set the global power level.

**C API:**

int32\_t gl\_ble\_set\_power(gl\_ble\_set\_power\_rsp\_t \*rsp, int32\_t power);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure that stores the current power. | |
| int32\_t | power | TX power in 0.1dBm steps, for example the value of 10 is 1dBm  and 55 is 5.5dBm | |

typedef struct {

int32\_t current\_power;

} gl\_ble\_set\_power\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| int32\_t | power | Actual adopted power level. | |

**CLI command:**

bletool set\_power 80

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| int32\_t | power | - | Power level |

## 2.4 listen

Listen to events generated from the BLE module.

**C API:**

int32\_t gl\_ble\_subscribe(gl\_ble\_cbs \*callback) ;

This function will subscribe events generate from BLE module. Note that it must be followed by uloop\_run(), it will continuously pass events to function callback.

typedef struct{

int32\_t (\*ble\_module\_event)(gl\_ble\_module\_event\_t event, gl\_ble\_module\_data\_t \*data);

int32\_t (\*ble\_gap\_event)(gl\_ble\_gap\_evrnt\_t event, gl\_ble\_gap\_data\_t \*data);

int32\_t (\*ble\_gatt\_event)(gl\_ble\_gatt\_event\_t event, gl\_ble\_gatt\_data\_t \*data);

} gl\_ble\_cbs;

### 2.4.1 int32\_t (\*ble\_module\_event)(gl\_ble\_module\_event\_t event, gl\_ble\_module\_data\_t \*data);

Indicates that the device has started and the radio is ready. This event carries the firmware build number and other software and hardware identification codes. User can get system boot event and use it in this callback. This callback will be called when module receive a system boot event.

/\* module callback event type \*/

typedef enum{

MODULE\_BLE\_SYSTEM\_BOOT\_EVT = 0,

MODULE\_EVT\_MAX,

} gl\_ble\_module\_event\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_module\_event\_t | | |
| Type | Name | Description | |
| enum | MODULE\_BLE\_SYSTEM\_BOOT\_EVT | BLE system event | |
| enum | MODULE\_EVT\_MAX | Event maximum | |

typedef union {

struct ble\_system\_boot\_data{

int32\_t major;

int32\_t minor;

int32\_t patch;

int32\_t build;

int32\_t bootloader;

int32\_t hw;

char ble\_hash[MAX\_HASH\_DATA\_LEN];

} system\_boot\_data;

} gl\_ble\_module\_data\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_module\_data\_t | | |
| Type | Name | Description | |
| int32\_t | major | Major release version | |
| int32\_t | minor | Minor release version | |
| Int | patch | Patch release number | |
| Int | build | Build number | |
| Int | bootloader | Bootloader version | |
| Int | hw | Hardware type | |
| char | ble\_hash | Version hash | |

### 2.4.2 int32\_t (\*ble\_gap\_event)(gl\_ble\_gap\_evrnt\_t event, gl\_ble\_gap\_data\_t \*data);

Receive BLE GAP event from the module. User can get GAP event data and use it in this callback. This callback will be called when module receive a GAP event.

/\* GAP BLE callback event type \*/

typedef enum{

GAP\_BLE\_SCAN\_RESULT\_EVT = 0,

GAP\_BLE\_UPDATE\_CONN\_EVT,

GAP\_BLE\_CONNECT\_EVT,

GAP\_BLE\_DISCONNECT\_EVT,

GAP\_BLE\_EVT\_MAX,

} gl\_ble\_gap\_evrnt\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_gap\_evrnt\_t | | |
| Type | Name | Description | |
| enum | GAP\_BLE\_SCAN\_RESULT\_EVT | Scan result event | |
| enum | GAP\_BLE\_UPDATE\_CONN\_EVT | Update connection event | |
| enum | GAP\_BLE\_CONNECT\_EVT | Connection event | |
| enum | GAP\_BLE\_DISCONNECT\_EVT | disconnection event | |
| enum | GAP\_BLE\_EVT\_MAX | Event maximum | |

#define DEVICE\_MAC\_LEN 6

#define BLE\_MAC\_LEN 18

typedef union {

struct ble\_scan\_result\_evt\_data {

char address[BLE\_MAC\_LEN];

gl\_ble\_address\_type\_t ble\_address\_type;

int32\_t packet\_type;

int32\_t rssi;

char ble\_adv[MAX\_ADV\_DATA\_LEN];

int32\_t bonding;

} scan\_rst;

struct ble\_update\_conn\_evt\_data {

uint8\_t address[DEVICE\_MAC\_LEN];

int32\_t interval;

int32\_t latency;

int32\_t timeout;

int32\_t security\_mode;

int32\_t txsize;

} update\_conn\_data;

struct ble\_connect\_open\_evt\_data {

char address[BLE\_MAC\_LEN];

gl\_ble\_address\_type\_t ble\_address\_type;

int32\_t conn\_role;

int32\_t connection;

int32\_t bonding;

int32\_t advertiser;

} connect\_open\_data;

struct ble\_disconnect\_evt\_data {

uint8\_t address[DEVICE\_MAC\_LEN];

int32\_t reason;

} disconnect\_data;

} gl\_ble\_gap\_data\_t;

// BLE device address type

typedef enum {

BLE\_ADDR\_TYPE\_PUBLIC = 0x00,

BLE\_ADDR\_TYPE\_RANDOM = 0x01,

BLE\_ANONYMOUS\_ADVERTISING = 0xff,

} gl\_ble\_address\_type\_t;

|  |  |  |
| --- | --- | --- |
| Scan\_rst | | |
| Type | Name | Description | |
| int32\_t | address | Bluetooth address of the remote device | |
| gl\_ble\_address\_type\_t | ble\_address\_type | Advertiser address type. Values:  0: Public address  1: Random address  255: No address provided (anonymous advertising) | |
| int32\_t | packet\_type | Bits 0..2: advertising packet type  000: Connectable scannable undirected advertising  001: Connectable undirected advertising  010: Scannable undirected advertising  011: Non-connectable non-scannable undirected advertising  100: Scan Response. Note that this is received only if the device is in active scan mode.  Bits 3..4: Reserved for the future  Bits 5..6: data completeness  00: Complete  01: Incomplete, more data to come in new events  10: Incomplete, data truncated, no more to come  Bit 7: legacy or extended advertising  0: Legacy advertising PDUs used  1: Extended advertising PDUs used | |
| int32\_t | rssi | Signal strength indicator (RSSI) in the latest received packet. Units: dBm. Range: -127 to +20 | |
| char | ble\_adv | Advertising or scan response data | |
| int32\_t | bonding | Bonding handle if the remote advertising device has previously bonded with the local device. Values:  0xff: No bonding  Other: Bonding handle | |

|  |  |  |
| --- | --- | --- |
| update\_conn\_data | | |
| Type | Name | Description | |
| int32\_t | connection | Connection handle | |
| int32\_t | interval | Connection interval. Time = Value x 1.25 ms | |
| int32\_t | latency | Slave latency (how many connection intervals the slave can skip) | |
| int32\_t | timeout | Supervision timeout. Time = Value x 10 ms | |
| int32\_t | security\_mode | Connection security mode | |
| int32\_t | txsize | Maximum Data Channel PDU Payload size that the controller can send in an air packet | |

|  |  |  |
| --- | --- | --- |
| connect\_open\_data | | |
| Type | Name | Description | |
| char | address | Remote device address | |
| gl\_ble\_address\_type\_t | ble\_address\_type | Remote device address type | |
| int32\_t | conn\_role | Device role in connection. Values:  0: Slave; 1: Master | |
| int32\_t | connection | Handle for new connection | |
| int32\_t | bonding | Bonding handle. Values:  0xff: No bonding; Other: Bonding handle | |
| int32\_t | advertiser | The local advertising set that this connection was opened to. Values:  0xff: Invalid value or not applicable. Ignore this field  Other: The advertising set handle | |

|  |  |  |
| --- | --- | --- |
| disconnect\_data | | |
| Type | Name | Description | |
| int32\_t | connection | Handle of the closed connection | |
| int32\_t | reason | Result code  0: success; Non-zero: an error has occurred  For other values see :  <https://docs.silabs.com/bluetooth/latest/error-codes> | |

### 2.4.3 int32\_t (\*ble\_gatt\_event)(gl\_ble\_gatt\_event\_t event, gl\_ble\_gatt\_data\_t \*data);

Receive BLE GATT event from the module. User can get GATT event data and use it in this callback. This callback will be called when module receive a GATT event.

/\* GATT BLE callback event type \*/

typedef enum

{

GATT\_BLE\_REMOTE\_NOTIFY\_EVT = 0,

GATT\_BLE\_REMOTE\_WRITE\_EVT,

GATT\_BLE\_REMOTE\_SET\_EVT,

GATT\_EVT\_MAX,

} gl\_ble\_gatt\_event\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_gatt\_evrnt\_t | | |
| Type | Name | Description | |
| enum | GATT\_BLE\_REMOTE\_NOTIFY\_EVT | Remote notify event | |
| enum | GATT\_BLE\_REMOTE\_WRITE\_EVT | Remote write event | |
| enum | GATT\_BLE\_REMOTE\_SET\_EVT | Remote set event | |
| enum | GATT\_EVT\_MAX | Event maximum | |

typedef union {

struct ble\_remote\_notify\_evt\_data {

uint8\_t address[DEVICE\_MAC\_LEN];

int32\_t characteristic;

int32\_t att\_opcode;

int32\_t offset;

char value[MAX\_VALUE\_DATA\_LEN];

} remote\_notify;

struct ble\_remote\_write\_evt\_data {

uint8\_t address[DEVICE\_MAC\_LEN];

int32\_t attribute;

int32\_t att\_opcode;

int32\_t offset;

char value[MAX\_VALUE\_DATA\_LEN];

} remote\_write;

struct ble\_remote\_set\_evt\_data {

uint8\_t address[DEVICE\_MAC\_LEN];

int32\_t characteristic;

int32\_t status\_flags;

int32\_t client\_config\_flags;

} remote\_set;

} gl\_ble\_gatt\_data\_t;

|  |  |  |
| --- | --- | --- |
| remote\_notify | | |
| Type | Name | Description | |
| int32\_t | connection | Connection handle | |
| int32\_t | characteristic | GATT characteristic handle | |
| int32\_t | att\_opcode | Attribute opcode, which indicates the GATT transaction used | |
| int32\_t | offset | Value offset | |
| char | value | Characteristic value | |

|  |  |  |
| --- | --- | --- |
| remote\_ write | | |
| Type | Name | Description | |
| int32\_t | connection | Connection handle | |
| int32\_t | attribute | Attribute handle | |
| int32\_t | att\_opcode | Attribute opcode, which indicates the GATT transaction used | |
| int32\_t | offset | Value offset | |
| char | value | Value | |

|  |  |  |
| --- | --- | --- |
| remote\_set | | |
| Type | Name | Description | |
| int32\_t | connection | Connection handle | |
| int32\_t | characteristic | GATT characteristic handle | |
| int32\_t | status\_flags | Describes whether Client Characteristic Configuration was changed or if a confirmation was received. | |
| int32\_t | 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. | |

int32\_t gl\_ble\_unsubscribe(void);

This function will unsubscribe the BLE events.

**CLI command:**

bletool listen

This command will not return. It will continuously print events generated from BLE module.

## 2.5 adv\_data

Act as BLE slave, set customized advertising data

**C API:**

int32\_t gl\_ble\_adv\_data(int32\_t flag, char \*data);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| int32\_t | flag | Adv data flag. This value selects if the data is intended for advertising packets, scan response packets or advertising packet in OTA.  • 0: Advertising packets • 1: Scan response packets • 2: OTA advertising packets • 4: OTA scan response packets | |
| string | data | Customized advertising data. Must be hexadecimal ASCII. Like “020106” | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool adv\_data –f 0 –v 020106

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| int32\_t | flag  -f | - | Adv data flag. |
| string | data  -v | - | Customized advertising data. |

## 2.6 adv

Set the advertising parameters and start advertising act as BLE slave.

**C API:**

int32\_t gl\_ble\_adv(int32\_t phys, int32\_t interval\_min, int32\_t interval\_max, int32\_t discover, int32\_t adv\_conn);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| int32\_t | phys | The PHY on which the advertising packets are transmitted on.  • 1: LE 1M PHY • 4: LE Coded PHY | |
| int32\_t | interval\_min | Minimum advertising interval. Value in units of 0.625 ms  • Range: 0x20 to 0xFFFF  • Time range: 20 ms to 40.96 s | |
| int32\_t | interval\_max | Maximum advertising interval. Value in units of 0.625 ms  • Range: 0x20 to 0xFFFF  • Time range: 20 ms to 40.96 s  • Note: interval\_max should be bigger than interval\_min | |
| int32\_t | discover | Define the discoverable mode.  • 0: Not discoverable  • 1: Discoverable using both limited and general discovery procedures  • 2: Discoverable using general discovery procedure  • 3: Device is not discoverable in either limited or generic discovery  procedure, but may be discovered by using the Observation procedure  • 4: Send advertising and/or scan response data defined by the user. The limited/general discoverable flags are defined by the user. | |
| int32\_t | adv\_conn | Connectable mode.  • 0: Non-connectable non-scannable  • 1: Directed connectable (RESERVED, DO NOT USE)  • 2: Undirected connectable scannable (This mode can only be used  in legacy advertising PDUs)  • 3: Undirected scannable (Non-connectable but responds to  scan requests)  • 4: Undirected connectable non-scannable. This mode can  only be used in extended advertising PDUs | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool adv

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| int32\_t | phys  -p | 1 | The PHY on which the advertising packets are transmitted on. |
| int32\_t | interval\_min  -n | 160  (100ms) | Minimum advertising interval. |
| int32\_t | interval\_max  -x | 160  (100ms) | Maximum advertising interval. |
| int32\_t | discover  -d | 2 | Discoverable mode. |
| int32\_t | connect  -c | 2 | Connectable mode. |

## 2.7 adv\_stop

Act as BLE slave, Stop advertising.

**C API:**

int32\_t gl\_ble\_stop\_adv(void);

No parameter.

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool adv\_stop

## 2.8 send\_notify

Act as BLE slave, send Notification to remote device.

**C API:**

int32\_t gl\_ble\_send\_notify(gl\_ble\_send\_notify\_rsp\_t \*rsp, char \*address, int32\_t char\_handle, char \*value);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure used for storing the length of the Notification. | |
| string | address | The MAC address of the remote device | |
| int32\_t | char\_handle | GATT characteristic handle | |
| string | value | Data value to be sent. | |

typedef struct {

int32\_t sent\_len;

} gl\_ble\_send\_notify\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_send\_notify\_rsp\_t | | |
| Type | Name | Description | |
| int32\_t | sent\_len | The length of notification to be sent | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool send\_notify

## 2.9 discovery

Act as master, set and start the BLE discovery.

**C API:**

int32\_t gl\_ble\_discovery(int32\_t phys, int32\_t interval, int32\_t window, int32\_t type, int32\_t mode);

Note that after call this function, BLE packets will be continuously pass to callback function registered by gl\_ble\_subscribe();

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| int32\_t | phys | The scanning PHY.  • 1: LE 1M PHY • 4: LE Coded PHY | |
| int32\_t | interval | Scan interval.  **•** Time = Value x 0.625 ms  **•** Range: 0x0004 to 0xFFFF  **•** Time Range: 2.5 ms to 40.96 s | |
| int32\_t | window | Scan window.  **•** Time = Value x 0.625 ms  **•** Range: 0x0004 to 0xFFFF  **•** Time Range: 2.5 ms to 40.96 s | |
| int32\_t | type | Scan type. Values:  **• 0:** Passive scanning  **• 1:** Active scanning  **•** In passive scanning mode, the device only listens to advertising packets and does not transmit packets.  **•** In active scanning mode, the device sends out a scan request  packet upon receiving an advertising packet from a remote device. Then, it listens to the scan response packet from the remote device | |
| int32\_t | mode | Bluetooth discovery Mode.  **• 0:** Discover only limited discoverable devices  **• 1:** Discover limited and generic discoverable devices  **• 2:** Discover all devices | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool discovery

Note that you have to using command “bletool listen*”* to receive BLE advertising packets after this command.

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| int32\_t | phys  -p | 1 | The scanning PHY. |
| int32\_t | interval  -i | 16  (10ms) | Scan interval. |
| int32\_t | window  -w | 16  (10ms) | Scan window. |
| int32\_t | type  -t | 0 | Scan type. |
| int32\_t | mode  -m | 1 | Bluetooth discovery Mode. |

## 2.10 stop

Act as master, stop discovery procedure.

**C API:**

int32\_t gl\_ble\_stop(void);

No parameter.

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool stop

## 2.11 connect

Act as master, start connect to a remote BLE device.

**C API:**

When this API is called, the struct pointer rsp will be populated.

int32\_t gl\_ble\_connect(gl\_ble\_connect\_rsp\_t \*rsp, char \*address, int32\_t address\_type, int32\_t phy);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure used for storing the connect parameters of the remote device. | |
| string | address | Remote BLE device address. Like “11:22:33:44:55:66” | |
| int32\_t | address\_type | Advertiser address type. Values:  **• 0:** Public address  **• 1:** Random address  **• 2:** Public identity address resolved by stack  **• 3:** Random identity address resolved by stack | |
| int32\_t | phy | The initiating PHY.  **• 1:** LE 1M PHY  **• 4:** LE Coded PHY | |

typedef struct {

uint8\_t address[DEVICE\_MAC\_LEN];

uint8\_t address\_type;

uint8\_t master;

uint8\_t bonding;

uint8\_t advertiser;

} gl\_ble\_connect\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_connect\_rsp\_t | | |
| Type | Name | Description | |
| uint8\_t | address | Remote BLE device address. Like “11:22:33:44:55:66” | |
| uint8\_t | address\_type | GATT characteristic handle | |
| uint8\_t | master | Data value to be sent. | |
| uint8\_t | bonding | Bonding handle if the remote advertising device has previously bonded with the local device. Values:  0xff: No bonding; Other: Bonding handle | |
| uint8\_t | advertiser | The local advertising set that this connection was opened to. Values:  0xff: Invalid value or not applicable. Ignore this field  Other: The advertising set handle | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| int32\_t | connection | Handle of new connection | |
| int32\_t | address | Remote device address | |
| int32\_t | address\_type | Remote device address type | |
| int32\_t | master | Device role in connection. Values:  **• 0:** Slave  **• 1:** Master | |
| int32\_t | bonding | Bonding handle if the remote advertising device has previously  bonded with the local device. Values:  **• 0xff:** No bonding  **• Other:** Bonding handle | |
| int32\_t | interval | Connection interval | |
| int32\_t | latency | Slave latency | |
| int32\_t | timeout | Connection timeout | |
| int32\_t | security\_mode | Connection security mode. Values:  **• 0:** No security  **• 1:** Unauthenticated pairing with encryption  **• 2:** Authenticated pairing with encryption  **• 3:** Authenticated Secure Connections pairing with encryption  using a 128-bit strength encryption key | |
| int32\_t | txsize | Maximum Data Channel PDU Payload size the controller can send in an air packet | |

**CLI command:**

bletool connect –a 11:22:33:44:55:66 –t 0

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address  -a | - | Remote BLE device address. |
| int32\_t | address\_type  -t | - | Advertiser address type. |
| int32\_t | phy  -p | 1 | The initiating PHY. |

## 2.12 disconnect

Act as master, disconnect with remote device.

**C API:**

int32\_t gl\_ble\_disconnect(char \*address);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| string | address | The MAC address of the remote device | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| string | address | The MAC address of the remote device | |
| int32\_t | reason | Connection disconnect reason | |

**CLI command:**

bletool disconnect 11:22:33:44:55:66

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address | - | The MAC address of the remote device |

## 2.13 get\_rssi

Act as master, get rssi of connection with remote device.

**C API:**

int32\_t gl\_ble\_get\_rssi(gl\_ble\_get\_rssi\_rsp\_t \*rsp, char \*address);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure that gets rssi | |
| string | address | The MAC address of the remote device | |

#define DEVICE\_MAC\_LEN 6

typedef struct {

uint8\_t address[DEVICE\_MAC\_LEN];

int32\_t rssi;

} gl\_ble\_get\_rssi\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_get\_rssi\_rsp\_t | | |
| Type | Name | Description | |
| uint8\_t | address | The MAC address of the remote device | |
| int32\_t | rssi | Signal strength indicator (RSSI) in the latest received packet. Units: dBm. Range: -127 to +20 | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| uint8 | address | The MAC address of the remote device | |
| int32\_t | rssi | Rssi of the specified connection (dBm) | |

**CLI command:**

bletool get\_rssi 1

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address  -a | - | The MAC address of the remote device |

## 2.14 get\_service

Act as master, get service list of a remote GATT server.

**C API:**

int32\_t gl\_ble\_get\_service(gl\_ble\_get\_service\_rsp\_t \*rsp, char \*address);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure that gets service list | |
| string | address | The MAC address of the remote device | |

#define DEVICE\_MAC\_LEN 6

typedef struct

{

uint8\_t address[DEVICE\_MAC\_LEN];

uint8\_t list\_len;

ble\_service\_list\_t list[LIST\_LENGTHE\_MAX];

} gl\_ble\_get\_service\_rsp\_t

#define  LIST\_LENGTHE\_MAX  16

typedef struct

{

int32\_t handle;

char uuid[UUID\_MAX];

} ble\_service\_list\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_get\_service\_rsp\_t | | |
| Type | Name | Description | |
| uint8\_t | address | The MAC address of the remote device | |
| uint8\_t | list\_len | Length of the service list | |
| ble\_service\_list\_t | list | Struct of the service list | |

|  |  |  |
| --- | --- | --- |
| ble\_service\_list\_t | | |
| Type | Name | Description | |
| int32\_t | handle | seivice handle | |
| char | uuid | UUID of characteristic | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success; None-zero means failed. | |
| uint8\_t | address | The MAC address of the remote device | |
| struct | service\_list | Array of service list | |

**CLI command:**

bletool get\_service 11:22:33:44:55:66

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address | - | The MAC address of the remote device |

## 2.15 get\_char

Act as master, Get characteristic list of a remote GATT server.

**C API:**

int32\_t gl\_ble\_get\_char(gl\_ble\_get\_char\_rsp\_t \*rsp, char \*address, int32\_t service\_handle);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure that gets characteristic list | |
| uint8\_t | address | The MAC address of the remote device | |
| int32\_t | service\_handle | service handle | |

#define DEVICE\_MAC\_LEN 6

typedef struct

{

uint8\_t address[DEVICE\_MAC\_LEN];

uint8\_t list\_len;

ble\_characteristic\_list\_t list[LIST\_LENGTHE\_MAX];

} gl\_ble\_get\_char\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_get\_char\_rsp\_t | | |
| Type | Name | Description | |
| uint8\_t | connection | characteristic handle | |
| uint8\_t | list\_len | Length of characteristic list | |
| ble\_characteristic\_list\_t | list | Struct of characteristic list | |

#define UUID\_MAX 32

typedef struct

{

int32\_t handle;

char uuid[UUID\_MAX];

uint8\_t properties;

} ble\_characteristic\_list\_t;

|  |  |  |
| --- | --- | --- |
| ble\_characteristic\_list\_t | | |
| Type | Name | Description | |
| int32\_t | handle | characteristic handle | |
| int32\_t | UUID | UUID of characteristic | |
| int32\_t | properties | Characteristic properties | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| int32\_t | connection | Connection handle | |
| jsonArray | characteristic\_list | Array of characteristics | |

**CLI command:**

bletool get\_char –a 11:22:33:44:55:66 –h 10789

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address  -a | - | The MAC address of the remote device |
| int32\_t | service\_handle  -h | - | Service handle |

## 2.16 set\_notify

Act as master, Enable or disable the notification or indication of a remote gatt server.

**C API:**

int32\_t gl\_ble\_set\_notify(char \*address, int32\_t char\_handle, int32\_t flag);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| string | address | The MAC address of the remote device | |
| int32\_t | char\_handle | Characteristic handle | |
| int32\_t | flag | Notification flag.  **• 0:** disable  **• 1:** notification  **• 2:** indication | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |

**CLI command:**

bletool set\_notify –a 11:22:33:44:55:66 –h 10789 –f 1

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address  -a | - | The MAC address of the remote device |
| int32\_t | char\_handle  -h | - | Characteristic handle |
| int32\_t | flag  -f | - | Notification flag. |

## 2.17 read\_value

Act as master, Read value of specified characteristic in a remote gatt server.

**C API:**

int32\_t gl\_ble\_read\_char(gl\_ble\_char\_read\_rsp\_t \*rsp, char \*address, int32\_t char\_handle);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A struct of read value response | |
| string | address | The MAC address of the remote device | |
| int32\_t | char\_handle | Characteristic handle | |

typedef struct {

    uint8\_t connection;

    int32\_t handle;

    uint8\_t att\_opcode;

    int32\_t offset;

    uint8\_t value[CHAR\_VALUE\_MAX];

} gl\_ble\_char\_read\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_char\_read\_rsp\_t | | |
| Type | Name | Description | |
| uint8\_t | connection | Connection handle | |
| int32\_t | handle | Characteristic handle | |
| uint8\_t | att\_opcode | Attribute opcode which informs the GATT transaction used. | |
| int32\_t | offset | Value offset | |
| uint8\_t | value | Characteristic value. In hexadecimal ASCII. Like “00560aff” | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success; None-zero means failed. | |
| int32\_t | connection | Connection handle | |
| int32\_t | char\_handle | Characteristic handle | |
| int32\_t | att\_opcode | Attribute opcode which informs the GATT transaction used. | |
| int32\_t | offset | Value offset | |
| string | value | Characteristic value. In hexadecimal ASCII. Like “00560aff” | |

**CLI command:**

bletool read\_value –a 11:22:33:44:55:66 –h 10789

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address  -a | - | The MAC address of the remote device |
| int32\_t | char\_handle  -h | - | Characteristic handle |

## 2.18 write\_value

Act as master, Write value to specified characteristic in a remote gatt server.

**C API:**

int32\_t gl\_ble\_write\_char(gl\_ble\_write\_char\_rsp\_t \*rsp, char \*address, int32\_t char\_handle, char \*value, int32\_t res);

|  |  |  |
| --- | --- | --- |
| Parameters | | |
| Type | Name | Description | |
| struct | rsp | A response structure that writes value to specified characteristic | |
| string | address | The MAC address of the remote device | |
| int32\_t | char\_handle | Characteristic handle | |
| string | value | Value to be written. Must be hexadecimal ASCII. Like “00010203” | |
| int32\_t | res | Response flag.  • 0: Write with no response  • 1: Write with response | |

typedef struct

{

int32\_t sent\_len;

} gl\_ble\_write\_char\_rsp\_t;

|  |  |  |
| --- | --- | --- |
| gl\_ble\_write\_char\_rsp\_t | | |
| Type | Name | Description | |
| int32\_t | sent\_len | Length of write value | |

|  |  |  |
| --- | --- | --- |
| Result | | |
| Type | Name | Description | |
| int32\_t | code | 0 means success;  None-zero means failed. | |
| int32\_t | sent\_len | Bytes be written successfully | |

**CLI command:**

bletool write\_value –a 11:22:33:44:55:66 –h 10789 –v 00000000 –r 0

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Default Value | Description |
| string | address  -a | - | The MAC address of the remote device |
| int32\_t | char\_handle  -h | - | Characteristic handle |
| string | value  -v |  | Value to be written |
| int32\_t | res  -r | 0 | Response flag |