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
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

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.

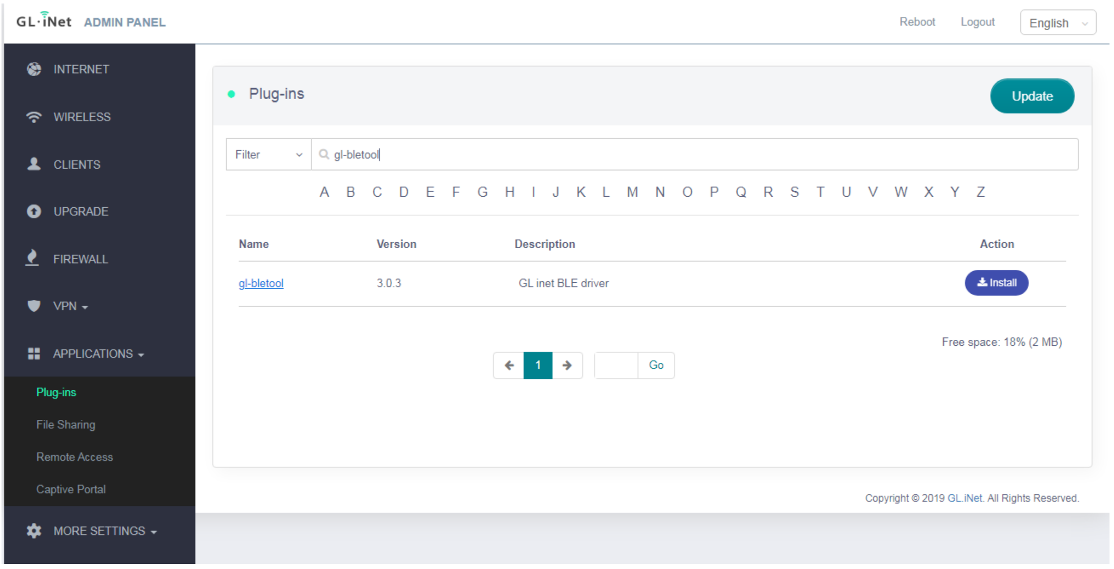
1.2 How to install

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".

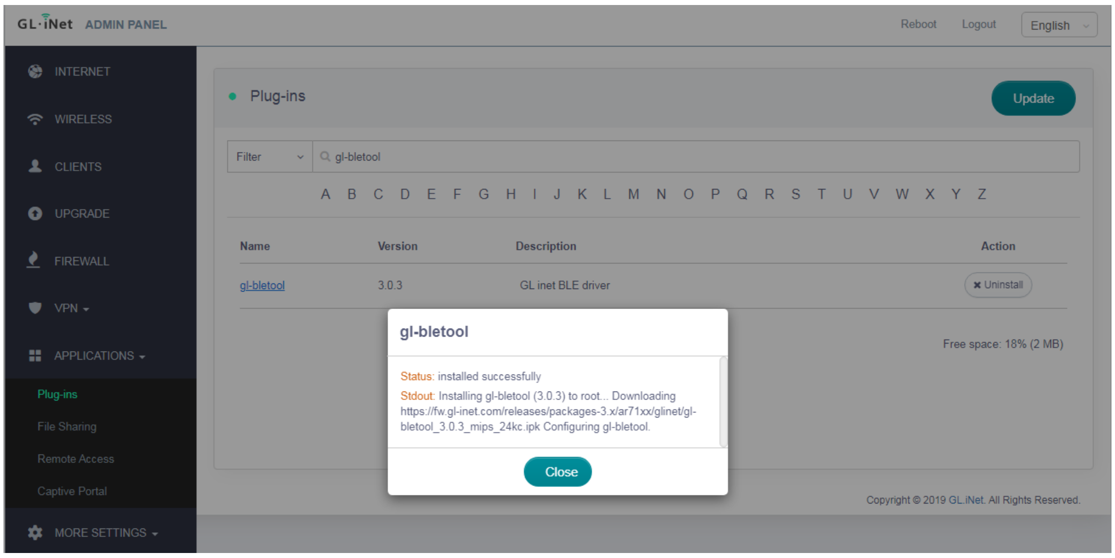


Search: gl-bletool

Name	Version	Description	Action
gl-bletool	3.0.3	GL inet BLE driver	Install

Free space: 18% (2 MB)

Copyright © 2019 GL.iNet. All Rights Reserved.



Name	Version	Description	Action
gl-bletool	3.0.3	GL inet BLE driver	✖ Uninstall

Free space: 18% (2 MB)

gl-bletool

Status: installed successfully

Stdout: Installing gl-bletool (3.0.3) to root... Downloading https://fw.gl-inet.com/releases/packages-3.x/ar71xx/glinet/gl-bletool_3.0.3_mips_24kc.ipk Configuring gl-bletool.

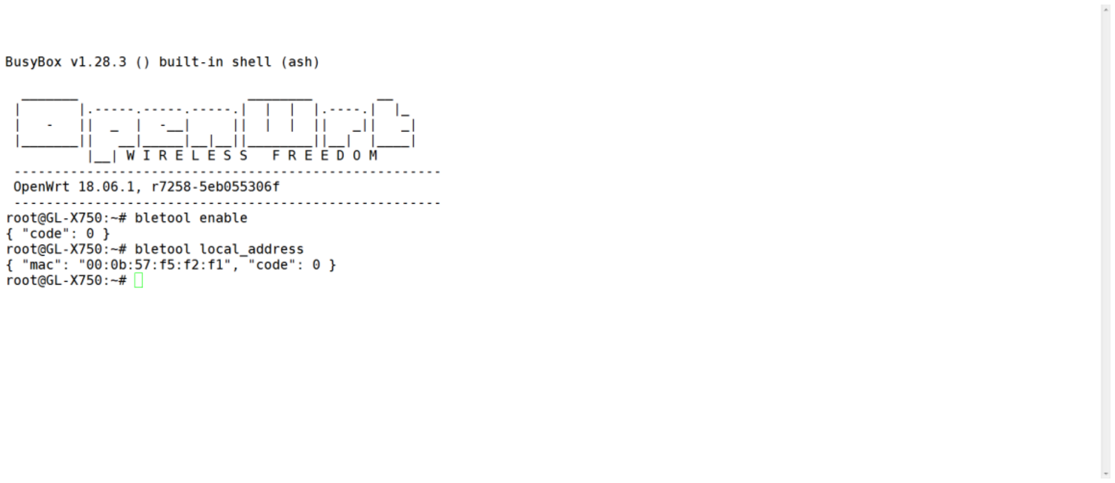
Copyright © 2019 GL.iNet. All Rights Reserved.

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.



```
BusyBox v1.28.3 () built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 18.06.1, r7258-5eb055306f
 -----------------------------------------------------
root@GL-X750:~# bletool enable
{ "code": 0 }
root@GL-X750:~# bletool local_address
{ "mac": "00:0b:57:f5:f2:f1", "code": 0 }
root@GL-X750:~# 
```

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:

```
int gl_ble_enable(int enable);
```

Parameters

Type	Name	Description
int	enable	0 means disable the BLE hardware; None-zero means enable the BLE hardware.

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.

CLI command:

```
bletool enable 1
```

Parameters

Type	Name	Default Value*	Description
int	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:

```
int gl_ble_get_mac(gl_ble_get_mac_rsp_t *rsp);
```

Parameters

Type	Name	Description
struct	rsp	A response structure that gets local Bluetooth MAC address

```
typedef struct {
   uint8_t addr[6];
} gl_ble_get_mac_rsp_t;
```

gl_ble_get_mac_rsp_t

Type	Name	Description
uint8_t	addr	The array of local Bluetooth MAC address

Result

Type	Name	Description
int	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:

```
int gl_ble_set_power(gl_ble_set_power_rsp_t *rsp, int power);
```

Parameters

Type	Name	Description
struct	rsp	A response structure that sets power
int	power	TX power in 0.1dBm steps, for example the value of 10 is 1dBm and 55 is 5.5dBm

```
typedef struct {
   int current_power;
} gl_ble_set_power_rsp_t;
```

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.
int	power	Actual adopted power level.

CLI command:

```
bletool set_power 80
```

Parameters

Type	Name	Default Value	Description
int	power	-	Power level

2.4 listen

Listen to events generated from the BLE module.

C API:

```
int 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{
   int (*ble_module_event)(gl_ble_module_event_t event, gl_ble_module_data_t *data);
   int (*ble_gap_event)(gl_ble_gap_evrnt_t event, gl_ble_gap_data_t *data);
   int (*ble_gatt_event)(gl_ble_gatt_event_t event, gl_ble_gatt_data_t *data);
} gl_ble_cbs;
```

2.4.1 int (*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{
        int major;
        int minor;
        int patch;
        int build;
        int bootloader;
        int hw;
        char ble_hash[MAX_HASH_DATA_LEN];
    } system_boot_data;
} gl_ble_module_data_t;
```

gl_ble_module_data_t

Type	Name	Description
int	major	Major release version
int	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 int (*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

```
typedef union {
    struct ble_scan_result_evt_data {
        char addr[BLE_MAC_LEN];
        gl_ble_addr_type_t ble_addr_type;
        int packet_type;
        int rssi;
        char ble_adv[MAX_ADV_DATA_LEN];
        int bonding;
    } scan_rst;

    struct ble_update_conn_evt_data {
        int connection;
        int interval;
        int latency;
        int timeout;
        int security_mode;
        int txsize;
    } update_conn_data;

    struct ble_connect_open_evt_data {
        char addr[BLE_MAC_LEN];
        gl_ble_addr_type_t ble_addr_type;
        int conn_role;
        int connection;
        int bonding;
        int advertiser;
    } connect_open_data;

   struct ble_disconnect_evt_data {
        int connection;
        int 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_addr_type_t;
```

Scan_rst

Type	Name	Description
int	addr	Bluetooth address of the remote device
gl_ble_addr_type_t	ble_addr_type	Advertiser address type. Values: 0: Public address 1: Random address 255: No address provided (anonymous advertising)
int	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
int	rssi	Signal strength indicator (RSSI) in the latest received packet. Units: dBm. Range: -127 to +20
char	ble_adv	Advertising or scan response data
int	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
int	connection	Connection handle
int	interval	Connection interval. Time = Value x 1.25 ms
int	latency	Slave latency (how many connection intervals the slave can skip)
int	timeout	Supervision timeout. Time = Value x 10 ms
int	security_mode	Connection security mode
int	txsize	Maximum Data Channel PDU Payload size that the controller can send in an air packet

connect_open_data

Type	Name	Description
char	addr	Remote device address
gl_ble_addr_type_t	ble_addr_type	Remote device address type
int	conn_role	Device role in connection. Values: 0: Slave; 1: Master
int	connection	Handle for new connection
int	bonding	Bonding handle. Values: 0xff: No bonding; Other: Bonding handle
int	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
int	connection	Handle of the closed connection
int	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 int (*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 {
        int connection;
        int characteristic;
        int att_opcode;
        int offset;
        char value[MAX_VALUE_DATA_LEN];

    } remote_notify;

    struct ble_remote_wirte_evt_data {
        int connection;
        int attribute;
        int att_opcode;
        int offset;
        char value[MAX_VALUE_DATA_LEN];
    } remote_write;

    struct ble_remote_set_evt_data  {
        int connection;
        int characteristic;
        int status_flags;
        int client_config_flags;
    } remote_set;
} gl_ble_gatt_data_t;
```

remote_notify

Type	Name	Description
int	connection	Connection handle
int	characteristic	GATT characteristic handle
int	att_opcode	Attribute opcode, which indicates the GATT transaction used
int	offset	Value offset
char	value	Characteristic value

remote_ write

Type	Name	Description
int	connection	Connection handle
int	attribute	Attribute handle
int	att_opcode	Attribute opcode, which indicates the GATT transaction used
int	offset	Value offset
char	value	Value

remote_set

Type	Name	Description
int	connection	Connection handle
int	characteristic	GATT characteristic handle
int	status_flags	Describes whether Client Characteristic Configuration was changed or if a confirmation was received.
int	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.

```
int 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:

```
int gl_ble_adv_data(int flag, char *data);
```

Parameters

Type	Name	Description
int	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
int	code	0 means success; None-zero means failed.

CLI command:

```
bletool adv_data -f 0 -v 020106
```

Parameters

Type	Name	Default Value	Description
int	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:

```
int gl_ble_adv(int phys, int interval_min, int interval_max, int discover, int
adv_conn);
```

Parameters

Type	Name	Description
int	phys	The PHY on which the advertising packets are transmitted on. • **1:** LE 1M PHY • **4:** LE Coded PHY
int	interval_min	Minimum advertising interval. Value in units of 0.625 ms • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s
int	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
int	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.
int	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
int	code	0 means success; None-zero means failed.

CLI command:

```
bletool adv
```

Parameters

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

2.7 adv_stop

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

C API:

```
int gl_ble_stop_adv(void);
```

No parameter.

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.

CLI command:

```
bletool adv_stop
```

2.8 send_notify

Act as GATT server, send Notification to remote device.

C API:

```
int gl_ble_send_notify(gl_ble_send_notify_rsp_t *rsp, char *address, int
char_handle, char *value);
```

Parameters

Type	Name	Description
struct	rsp	A response structure that sends notification
string	address	The MAC address of the remote device
int	char_handle	GATT characteristic handle
string	value	Data value to be sent.

```
typedef struct {
    int sent_len;
} gl_ble_send_notify_rsp_t;
```

gl_ble_send_notify_rsp_t

Type	Name	Description
int	sent_len	The length of notification to be sent

Result

Type	Name	Description
int	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:

```
int gl_ble_discovery(int phys, int interval, int window, int type, int 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
int	phys	The scanning PHY. • 1: LE 1M PHY • 4: LE Coded PHY
int	interval	Scan interval. • Time = Value x 0.625 ms • Range: 0x0004 to 0xFFFF • Time Range: 2.5 ms to 40.96 s
int	window	Scan window. • Time = Value x 0.625 ms • Range: 0x0004 to 0xFFFF • Time Range: 2.5 ms to 40.96 s
int	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
int	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
int	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
int	phys -p	1	The scanning PHY.
int	interval -i	16 (10ms)	Scan interval.
int	window -w	16 (10ms)	Scan window.
int	type -t	0	Scan type.
int	mode -m	1	Bluetooth discovery Mode.

2.10 stop

Act as master, stop discovery procedure.

C API:

```
int gl_ble_stop(void);
```

No parameter.

Result

Type	Name	Description
int	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.

```
int gl_ble_connect(gl_ble_connect_rsp_t *rsp, char *address, int address_type, int phy);
```

Parameters

Type	Name	Description
struct	rsp	A response structure that creates connection
string	address	Remote BLE device address. Like “11:22:33:44:55:66”
int	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
int	phy	The initiating PHY. • **1:** LE 1M PHY • **4:** LE Coded PHY

```
typedef struct {
    uint8_t connection;
    uint8_t addr[6];
    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	connection	Connection handle
uint8_t	addr	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
int	code	0 means success; None-zero means failed.
int	connection	Handle of new connection
int	address	Remote device address
int	address_type	Remote device address type
int	master	Device role in connection. Values: • **0:** Slave • **1:** Master
int	bonding	Bonding handle if the remote advertising device has previously bonded with the local device. Values: • **0xff:** No bonding • **Other:** Bonding handle
int	interval	Connection interval
int	latency	Slave latency
int	timeout	Connection timeout
int	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
int	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.
int	address_type -t	-	Advertiser address type.
int	phy -p	1	The initiating PHY.

2.12 disconnect

Act as master, disconnect with remote device.

C API:

```
int gl_ble_disconnect(char *address);
```

Parameters

Type	Name	Description
string	address	The MAC address of the remote device

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.
string	address	The MAC address of the remote device
int	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:

```
int 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 addr[DEVICE_MAC_LEN];
    int 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
int	rssi	Signal strength indicator (RSSI) in the latest received packet. Units: dBm. Range: -127 to +20

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.
uint8	address	The MAC address of the remote device
int	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:

```
int 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 addr[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
{
   int 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

int	handle	seivice handle
char	uuid	UUID of characteristic

Result

Type	Name	Description
int	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:

```
int gl_ble_get_char(gl_ble_get_char_rsp_t *rsp, char *address, int 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
int	service_handle	service handle

```
#define DEVICE_MAC_LEN            6

typedef struct
{
    uint8_t addr[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
{
   int handle;
   char uuid[UUID_MAX];
   uint8_t properties;
} ble_characteristic_list_t;
```

ble_characteristic_list_t

Type	Name	Description
int	handle	characteristic handle
int	UUID	UUID of characteristic
int	properties	Characteristic properties

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.
int	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
int	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:

```
int gl_ble_set_notify(char *address, int char_handle, int flag);
```

Parameters

Type	Name	Description
string	address	The MAC address of the remote device
int	char_handle	Characteristic handle
int	flag	Notification flag. • **0:** disable • **1:** notification • **2:** indication

Result

Type	Name	Description
int	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
int	char_handle -h	-	Characteristic handle
int	flag	-	Notification flag.

	-f		

2.17 read_value

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

C API:

```
int gl_ble_read_char(gl_ble_char_read_rsp_t *rsp, char *address, int char_handle);
```

Parameters

Type	Name	Description
struct	rsp	A struct of read value response
string	address	The MAC address of the remote device
int	char_handle	Characteristic handle

```
typedef struct {
    uint8_t connection;
    int handle;
    uint8_t att_opcode;
    int 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
int	handle	Characteristic handle
uint8_t	att_opcode	Attribute opcode which informs the GATT transaction used.
int	offset	Value offset
uint8_t	value	Characteristic value. In hexadecimal ASCII. Like “00560aff”

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.
int	connection	Connection handle
int	char_handle	Characteristic handle
int	att_opcode	Attribute opcode which informs the GATT transaction used.

int	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
int	char_handle -h	-	Characteristic handle

2.18 write_value

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

C API:

```
int gl_ble_write_char(gl_ble_write_char_rsp_t *rsp, char *address, int char_handle,
char *value, int 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
int	char_handle	Characteristic handle
string	value	Value to be written. Must be hexadecimal ASCII. Like “00010203”
int	res	Response flag. • **0:** Write with no response • **1:** Write with response

```
typedef struct
{
   int sent_len;
} gl_ble_write_char_rsp_t;
```

gl_ble_write_char_rsp_t

Type	Name	Description

int	sent_len	Length of write value

Result

Type	Name	Description
int	code	0 means success; None-zero means failed.
int	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
int	char_handle -h	-	Characteristic handle
string	value -v		Value to be written
int	res -r	0	Response flag