GATT SERVER API

Overview

Instructions

Application Example

Check bluetooth/bluedroid/ble folder in ESP-IDF examples, which contains the following demos and their tutorials:

- This is a GATT sever demo and its tutorial. This demo creates a GATT service with an attribute table, which releases the user from adding attributes one by one. This is the recommended method of adding attributes.
    - bluetooth/bluedroid/ble/gatt_server_service_table
    - GATT Server Service Table Example Walkthrough
- This is a GATT server demo and its tutorial. This demo creates a GATT service by adding attributes one by one as defined by Bluedroid. The recommended method of adding attributes is presented in example above.
    - bluetooth/bluedroid/ble/gatt_server
    - GATT Server Example Walkthrough
- This is a BLE SPP-Like demo. This demo, which acts as a GATT server, can receive data from UART and then send the data to the peer device automatically.
    - bluetooth/bluedroid/ble/ble_spp_server

API Reference

Header File

- bt/host/bluedroid/api/include/api/esp_gatts_api.h

Functions

esp_err_t **esp_ble_gatts_register_callback**(esp_gatts_cb_t *callback*)

This function is called to register application callbacks with BTA GATTS module.

Return

- ESP_OK : success

esp_err_t **esp_ble_gatts_app_register**(uint16_t *app_id*)

This function is called to register application identifier.

Return

- ESP_OK : success
- other : failed

esp_err_t **esp_ble_gatts_app_unregister**(esp_gatt_if_t *gatts_if*)

unregister with GATT Server.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface

esp_err_t **esp_ble_gatts_create_service**(esp_gatt_if_t *gatts_if*, esp_gatt_srvc_id_t * *service_id*, uint16_t *num_handle*)

Create a service. When service creation is done, a callback event BTA_GATTS_CREATE_SRVC_EVT is called to report status and service ID to the profile. The service ID obtained in the callback function needs to be used when adding included service and characteristics/descriptors into the service.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface
- `[in] service_id` : service ID.
- `[in] num_handle` : number of handle requested for this service.

esp_err_t **esp_ble_gatts_create_attr_tab**(const esp_gatts_attr_db_t * *gatts_attr_db*, esp_gatt_if_t *gatts_if*, uint8_t *max_nb_attr*, uint8_t *srvc_inst_id*)

Create a service attribute tab.

Return

- ESP_OK : success

Parameters

- `[in] gatts_attr_db` : the pointer to the service attr tab
- `[in] gatts_if` : GATT server access interface
- `[in] max_nb_attr` : the number of attribute to be added to the service database.
- `[in] srvc_inst_id` : the instance id of the service

---

esp_err_t **esp_ble_gatts_add_included_service**(uint16_t *service_handle*, uint16_t *included_service_handle*)

This function is called to add an included service. This function have to be called between 'esp_ble_gatts_create_service' and 'esp_ble_gatts_add_char'. After included service is included, a callback event BTA_GATTS_ADD_INCL_SRVC_EVT is reported the included service ID.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] service_handle` : service handle to which this included service is to be added.
- `[in] included_service_handle` : the service ID to be included.

---

esp_err_t **esp_ble_gatts_add_char**(uint16_t *service_handle*, esp_bt_uuid_t * *char_uuid*, esp_gatt_perm_t *perm*, esp_gatt_char_prop_t *property*, esp_attr_value_t * *char_val*, esp_attr_control_t * *control*)

This function is called to add a characteristic into a service.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] service_handle` : service handle to which this included service is to be added.
- `[in] char_uuid` : : Characteristic UUID.
- `[in] perm` : : Characteristic value declaration attribute permission.
- `[in] property` : : Characteristic Properties
- `[in] char_val` : : Characteristic value
- `[in] control` : : attribute response control byte

---

esp_err_t **esp_ble_gatts_add_char_descr**(uint16_t *service_handle*, esp_bt_uuid_t * *descr_uuid*,

esp_gatt_perm_t *perm*, esp_attr_value_t * *char_descr_val*, esp_attr_control_t * *control*)

This function is called to add characteristic descriptor. When it's done, a callback event BTA_GATTS_ADD_DESCR_EVT is called to report the status and an ID number for this descriptor.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] service_handle` : service handle to which this characteristic descriptor is to be added.
- `[in] perm` : descriptor access permission.
- `[in] descr_uuid` : descriptor UUID.
- `[in] char_descr_val` : : Characteristic descriptor value
- `[in] control` : : attribute response control byte

---

esp_err_t **esp_ble_gatts_delete_service**(uint16_t *service_handle*)

This function is called to delete a service. When this is done, a callback event BTA_GATTS_DELETE_EVT is report with the status.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] service_handle` : service_handle to be deleted.

---

esp_err_t **esp_ble_gatts_start_service**(uint16_t *service_handle*)

This function is called to start a service.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] service_handle` : the service handle to be started.

---

esp_err_t **esp_ble_gatts_stop_service**(uint16_t *service_handle*)

This function is called to stop a service.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] service_handle` : - service to be topped.

---

esp_err_t **esp_ble_gatts_send_indicate**(esp_gatt_if_t *gatts_if*, uint16_t *conn_id*, uint16_t *attr_handle*, uint16_t *value_len*, uint8_t **value*, bool *need_confirm*)

Send indicate or notify to GATT client. Set param need_confirm as false will send notification, otherwise indication.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface
- `[in] conn_id` : - connection id to indicate.
- `[in] attr_handle` : - attribute handle to indicate.
- `[in] value_len` : - indicate value length.
- `[in] value` : value to indicate.
- `[in] need_confirm` : - Whether a confirmation is required. false sends a GATT notification, true sends a GATT indication.

---

esp_err_t **esp_ble_gatts_send_response**(esp_gatt_if_t *gatts_if*, uint16_t *conn_id*, uint32_t *trans_id*, esp_gatt_status_t *status*, esp_gatt_rsp_t * *rsp*)

This function is called to send a response to a request.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface
- `[in] conn_id` : - connection identifier.
- `[in] trans_id` : - transfer id
- `[in] status` : - response status

`[in] esp`

---

esp_err_t **esp_ble_gatts_set_attr_value**(uint16_t *attr_handle*, uint16_t *length*, *const* uint8_t **value*)

This function is called to set the attribute value by the application.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] attr_handle` : the attribute handle which to be set
- `[in] length` : the value length
- `[in] value` : the pointer to the attribute value

---

esp_gatt_status_t **esp_ble_gatts_get_attr_value**(uint16_t *attr_handle*, uint16_t **length*, *const* uint8_t ***value*)

Retrieve attribute value.

Return

- ESP_GATT_OK : success
- other : failed

Parameters

- `[in] attr_handle` : Attribute handle.
- `[out] length` : pointer to the attribute value length
- `[out] value` : Pointer to attribute value payload, the value cannot be modified by user

---

esp_err_t **esp_ble_gatts_open**(esp_gatt_if_t *gatts_if*, esp_bd_addr_t *remote_bda*, bool *is_direct*)

Open a direct open connection or add a background auto connection.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface
- `[in] remote_bda` : remote device bluetooth device address.
- `[in] is_direct` : direct connection or background auto connection

esp_err_t **esp_ble_gatts_close**(esp_gatt_if_t *gatts_if*, uint16_t *conn_id*)

Close a connection a remote device.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface
- `[in] conn_id` : connection ID to be closed.

---

esp_err_t **esp_ble_gatts_send_service_change_indication**(esp_gatt_if_t *gatts_if*, esp_bd_addr_t *remote_bda*)

Send service change indication.

Return

- ESP_OK : success
- other : failed

Parameters

- `[in] gatts_if` : GATT server access interface
- `[in] remote_bda` : remote device bluetooth device address. If remote_bda is NULL then it will send service change indication to all the connected devices and if not then to a specific device

Unions

union `esp_ble_gatts_cb_param_t`

#include <esp_gatts_api.h>
Gatt server callback parameters union.

Public Members

struct esp_ble_gatts_cb_param_t::gatts_reg_evt_param `reg`

Gatt server callback param of ESP_GATTS_REG_EVT

struct esp_ble_gatts_cb_param_t::gatts_read_evt_param `read`

Gatt server callback param of ESP_GATTS_READ_EVT

struct esp_ble_gatts_cb_param_t::gatts_write_evt_param `write`

Gatt server callback param of ESP_GATTS_WRITE_EVT

struct esp_ble_gatts_cb_param_t::esp_ble_gatts_cb_param_t::gatts_exec_write_evt_param `exec_write`

Gatt server callback param of ESP_GATTS_EXEC_WRITE_EVT

struct esp_ble_gatts_cb_param_t::gatts_mtu_evt_param `mtu`

Gatt server callback param of ESP_GATTS_MTU_EVT

struct esp_ble_gatts_cb_param_t::gatts_conf_evt_param `conf`

Gatt server callback param of ESP_GATTS_CONF_EVT (confirm)

struct esp_ble_gatts_cb_param_t::gatts_create_evt_param `create`

Gatt server callback param of ESP_GATTS_CREATE_EVT

struct esp_ble_gatts_cb_param_t::gatts_add_incl_srvc_evt_param `add_incl_srvc`

Gatt server callback param of ESP_GATTS_ADD_INCL_SRVC_EVT

struct esp_ble_gatts_cb_param_t::gatts_add_char_evt_param `add_char`

Gatt server callback param of ESP_GATTS_ADD_CHAR_EVT

struct esp_ble_gatts_cb_param_t::gatts_add_char_descr_evt_param `add_char_descr`

Gatt server callback param of ESP_GATTS_ADD_CHAR_DESCR_EVT

struct esp_ble_gatts_cb_param_t::gatts_delete_evt_param `del`

Gatt server callback param of ESP_GATTS_DELETE_EVT

struct esp_ble_gatts_cb_param_t::gatts_start_evt_param `start`

Gatt server callback param of ESP_GATTS_START_EVT

struct esp_ble_gatts_cb_param_t::gatts_stop_evt_param `stop`

Gatt server callback param of ESP_GATTS_STOP_EVT

struct esp_ble_gatts_cb_param_t::gatts_connect_evt_param `connect`

Gatt server callback param of ESP_GATTS_CONNECT_EVT

struct esp_ble_gatts_cb_param_t::gatts_disconnect_evt_param `disconnect`

Gatt server callback param of ESP_GATTS_DISCONNECT_EVT

struct esp_ble_gatts_cb_param_t::gatts_open_evt_param `open`

Gatt server callback param of ESP_GATTS_OPEN_EVT

struct esp_ble_gatts_cb_param_t::esp_ble_gatts_cb_param_t::gatts_cancel_open_evt_param `cancel_open`

Gatt server callback param of ESP_GATTS_CANCEL_OPEN_EVT

struct esp_ble_gatts_cb_param_t::gatts_close_evt_param `close`

Gatt server callback param of ESP_GATTS_CLOSE_EVT

struct esp_ble_gatts_cb_param_t::gatts_congest_evt_param `congest`

Gatt server callback param of ESP_GATTS_CONGEST_EVT

struct esp_ble_gatts_cb_param_t::gatts_rsp_evt_param `rsp`

Gatt server callback param of ESP_GATTS_RESPONSE_EVT

struct esp_ble_gatts_cb_param_t::gatts_add_attr_tab_evt_param `add_attr_tab`

Gatt server callback param of ESP_GATTS_CREAT_ATTR_TAB_EVT

struct esp_ble_gatts_cb_param_t::gatts_set_attr_val_evt_param `set_attr_val`

Gatt server callback param of ESP_GATTS_SET_ATTR_VAL_EVT

struct esp_ble_gatts_cb_param_t::gatts_send_service_change_evt_param `service_change`

Gatt server callback param of ESP_GATTS_SEND_SERVICE_CHANGE_EVT

---

struct **gatts_add_attr_tab_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_CREAT_ATTR_TAB_EVT.

Public Members

esp_gatt_status_t **status**

Operation status

esp_bt_uuid_t **svc_uuid**

Service uuid type

uint8_t svc_inst_id

Service id

uint16_t num_handle

The number of the attribute handle to be added to the gatts database

uint16_t *handles

The number to the handles

struct `gatts_add_char_descr_evt_param`

#include <esp_gatts_api.h>

ESP_GATTS_ADD_CHAR_DESCR_EVT.

Public Members

esp_gatt_status_t **status**

Operation status

uint16_t attr_handle

Descriptor attribute handle

uint16_t service_handle

Service attribute handle

esp_bt_uuid_t **descr_uuid**

Characteristic descriptor uuid

struct `gatts_add_char_evt_param`

#include <esp_gatts_api.h>

ESP_GATTS_ADD_CHAR_EVT.

Public Members

esp_gatt_status_t **status**

Operation status

uint16_t attr_handle

Characteristic attribute handle

uint16_t service_handle

Service attribute handle

esp_bt_uuid_t **char_uuid**

Characteristic uuid

struct `gatts_add_incl_srvc_evt_param`

#include <esp_gatts_api.h>

ESP_GATTS_ADD_INCL_SRVC_EVT.

Public Members

esp_gatt_status_t **status**

Operation status

uint16_t attr_handle

Included service attribute handle

uint16_t service_handle

Service attribute handle

---

struct **gatts_cancel_open_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_CANCEL_OPEN_EVT.

Public Members

esp_gatt_status_t status

Operation status

---

struct **gatts_close_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_CLOSE_EVT.

Public Members

esp_gatt_status_t status

Operation status

uint16_t conn_id

Connection id

---

struct **gatts_conf_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_CONF_EVT.

Public Members

esp_gatt_status_t status

Operation status

uint16_t conn_id

Connection id

uint16_t handle

attribute handle

uint16_t len

The indication or notification value length, len is valid when send notification or

uint8_t *value

The indication or notification value , value is valid when send notification or indication failed

---

struct **gatts_congest_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_LISTEN_EVT.

ESP_GATTS_CONGEST_EVT

Public Members

uint16_t conn_id

Connection id

bool congested

Congested or not

---

struct **gatts_connect_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_CONNECT_EVT.

Public Members

uint16_t conn_id

Connection id

esp_bd_addr_t remote_bda

Remote bluetooth device address

esp_gatt_conn_params_t conn_params

current Connection parameters

---

struct **gatts_create_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_UNREG_EVT.

ESP_GATTS_CREATE_EVT

Public Members

esp_gatt_status_t status

Operation status

uint16_t service_handle

Service attribute handle

esp_gatt_srvc_id_t **service_id**

Service id, include service uuid and other information

---

struct **gatts_delete_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_DELETE_EVT.

Public Members

esp_gatt_status_t **status**

Operation status

uint16_t service_handle

Service attribute handle

---

struct **gatts_disconnect_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_DISCONNECT_EVT.

Public Members

uint16_t conn_id

Connection id

esp_bd_addr_t **remote_bda**

Remote bluetooth device address

esp_gatt_conn_reason_t **reason**

Indicate the reason of disconnection

---

struct **gatts_exec_write_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_EXEC_WRITE_EVT.

Public Members

uint16_t conn_id

Connection id

uint32_t trans_id

Transfer id

esp_bd_addr_t **bda**

The bluetooth device address which been written

uint8_t exec_write_flag

Execute write flag

---

struct **gatts_mtu_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_MTU_EVT.

Public Members

uint16_t conn_id

Connection id

uint16_t mtu

MTU size

---

struct **gatts_open_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_OPEN_EVT.

Public Members

esp_gatt_status_t **status**

Operation status

---

struct **gatts_read_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_READ_EVT.

Public Members

uint16_t conn_id

Connection id

uint32_t trans_id

Transfer id

esp_bd_addr_t **bda**

The bluetooth device address which been read

uint16_t handle

The attribute handle

uint16_t offset

Offset of the value, if the value is too long

bool is_long

The value is too long or not

bool need_rsp

The read operation need to do response

---

struct **gatts_reg_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_REG_EVT.

Public Members

esp_gatt_status_t status

Operation status

uint16_t app_id

Application id which input in register API

---

struct **gatts_rsp_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_RESPONSE_EVT.

Public Members

esp_gatt_status_t status

Operation status

uint16_t handle

Attribute handle which send response

---

struct **gatts_send_service_change_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_SEND_SERVICE_CHANGE_EVT.

Public Members

esp_gatt_status_t status

Operation status

---

struct **gatts_set_attr_val_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_SET_ATTR_VAL_EVT.

Public Members

uint16_t srvc_handle

The service handle

uint16_t attr_handle

The attribute handle

esp_gatt_status_t status

Operation status

---

struct **gatts_start_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_START_EVT.

Public Members

esp_gatt_status_t status

Operation status

uint16_t service_handle

Service attribute handle

---

struct **gatts_stop_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_STOP_EVT.

Public Members

esp_gatt_status_t status

Operation status

uint16_t service_handle

Service attribute handle

---

struct **gatts_write_evt_param**

#include <esp_gatts_api.h>

ESP_GATTS_WRITE_EVT.

Public Members

uint16_t conn_id

Connection id

uint32_t trans_id

Transfer id

esp_bd_addr_t **bda**

The bluetooth device address which been written

uint16_t handle

The attribute handle

uint16_t offset

Offset of the value, if the value is too long

bool need_rsp

The write operation need to do response

bool is_prep

This write operation is prepare write

uint16_t len

The write attribute value length

uint8_t *value

The write attribute value

Macros

`ESP_GATT_PREP_WRITE_CANCEL`

Prepare write flag to indicate cancel prepare write

`ESP_GATT_PREP_WRITE_EXEC`

Prepare write flag to indicate execute prepare write

Type Definitions

typedef void (***esp_gatts_cb_t**)(esp_gatts_cb_event_t event, esp_gatt_if_t gatts_if, esp_ble_gatts_cb_param_t *param)

GATT Server callback function type.

Parameters

- `event` : : Event type
- `gatts_if` : : GATT server access interface, normally different gatts_if correspond to different profile

Enumerations

enum `esp_gatts_cb_event_t`

GATT Server callback function events.

Values:

`ESP_GATTS_REG_EVT` = 0

When register application id, the event comes

`ESP_GATTS_READ_EVT` = 1

When gatt client request read operation, the event comes

`ESP_GATTS_WRITE_EVT` = 2

When gatt client request write operation, the event comes

`ESP_GATTS_EXEC_WRITE_EVT` = 3

When gatt client request execute write, the event comes

`ESP_GATTS_MTU_EVT` = 4

When set mtu complete, the event comes

`ESP_GATTS_CONF_EVT` = 5

When receive confirm, the event comes

`ESP_GATTS_UNREG_EVT` = 6

When unregister application id, the event comes

`ESP_GATTS_CREATE_EVT` = 7

When create service complete, the event comes

`ESP_GATTS_ADD_INCL_SRVC_EVT` = 8

When add included service complete, the event comes

`ESP_GATTS_ADD_CHAR_EVT` = 9

When add characteristic complete, the event comes

`ESP_GATTS_ADD_CHAR_DESCR_EVT` = 10

When add descriptor complete, the event comes

`ESP_GATTS_DELETE_EVT` = 11

When delete service complete, the event comes

`ESP_GATTS_START_EVT` = 12

When start service complete, the event comes

`ESP_GATTS_STOP_EVT` = 13

When stop service complete, the event comes

`ESP_GATTS_CONNECT_EVT` = 14

When gatt client connect, the event comes

`ESP_GATTS_DISCONNECT_EVT` = 15

When gatt client disconnect, the event comes

`ESP_GATTS_OPEN_EVT` = 16

When connect to peer, the event comes

`ESP_GATTS_CANCEL_OPEN_EVT` = 17

When disconnect from peer, the event comes

`ESP_GATTS_CLOSE_EVT` = 18

When gatt server close, the event comes

`ESP_GATTS_LISTEN_EVT` = 19

When gatt listen to be connected the event comes

`ESP_GATTS_CONGEST_EVT` = 20

When congest happen, the event comes

`ESP_GATTS_RESPONSE_EVT` = 21

When gatt send response complete, the event comes

`ESP_GATTS_CREAT_ATTR_TAB_EVT` = 22

When gatt create table complete, the event comes

`ESP_GATTS_SET_ATTR_VAL_EVT` = 23

When gatt set attr value complete, the event comes

```
ESP_GATTS_SEND_SERVICE_CHANGE_EVT = 24
```

When gatt send service change indication complete, the event comes