IoT SDK Bluetooth Audio Sink Developer's Guide

Version: 0.1

Release date: 2022-01-28

Version History

Version	Date	Description
0.1	2022-01-28	Initial Draft release

Table of Contents

List of Figures

List of Tables

1 Overview

MT793X development platform provides Bluetooth and Bluetooth Low Energy (LE) connectivity support for IoT devices. Bluetooth standard offers basic rate (BR) or enhanced data rate (EDR) and Bluetooth Low Energy (LE) support. Devices that can support BR/EDR and Bluetooth LE are referred to as dual-mode devices. Typically, in a Bluetooth system, a mobile phone or laptop computer acts as a dual-mode device. Devices that only support Bluetooth LE are referred to as single-mode devices where the low power consumption is the primary concern for application development, such as those that run on coin cell batteries.

This document guides you through:

- How to implement your application for Bluetooth music.
- Architecture of Bluetooth music system and folders.
- Introduction of each module related to Bluetooth music.

1.1 System Architecture

As shown in Figure 1, the application layer of MediaTek is composed of three kinds of modules: event senders, APPs, and Config. Event senders send UI shell events to APPs. APPs receive the events and take actions to complete the applications. Configs manage provide key actions and project configuration.

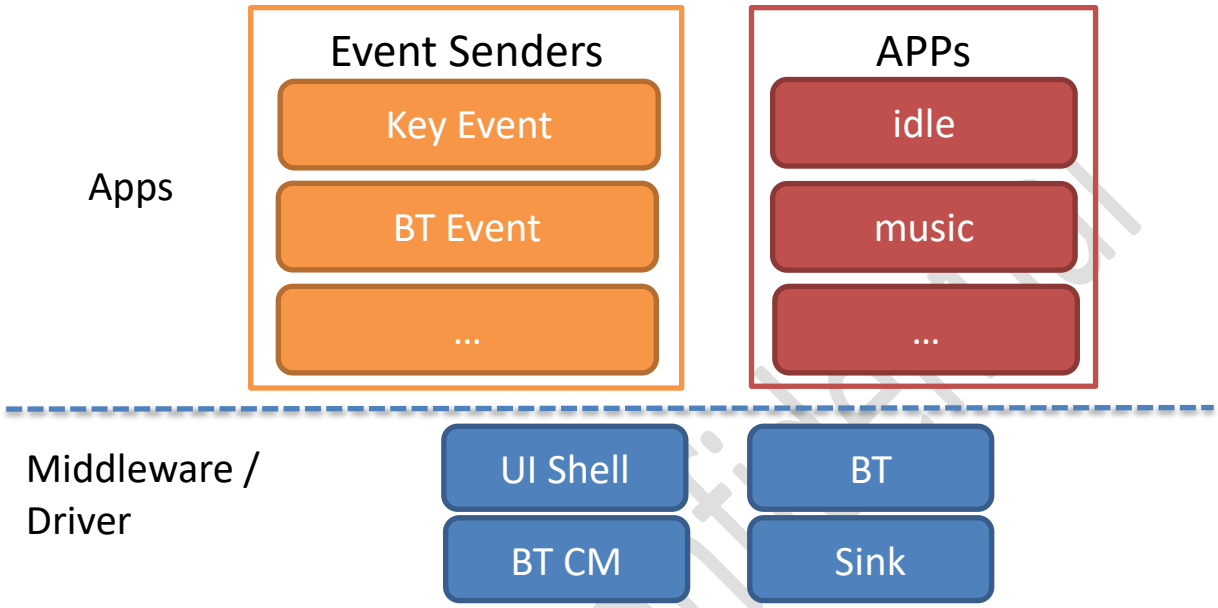

Figure 1 System Architecture

Event senders include key events, interaction events, and others. These modules register callbacks in middleware or hardware abstraction layer modules. When the callbacks are executed, the events are sent to APPs.

Take BT events as an example:

- The event sender sends an event to UI shell when called. After receiving the event sent by the key event sender, the UI shell sends it to APPs.

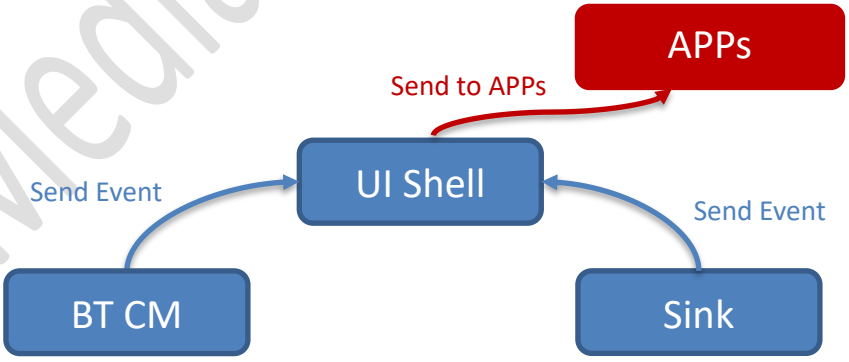

Figure 2 BT Event Sender

Multiple folders named "app_***" are APPs, which implement UI logic. An APP can send events to another one for the request of taking some actions, or notifying the happening of some events.

1.2 Folder Structure

The folders are listed below for your reference. For the detailed design of MediaTek IoT SDK application, refer to the source code.

- APPs
  /project/<board>/apps/<project>/src/apps/app_*
- Configs
  /project/<board>/apps/<project>/src/apps/config
- Event sender
  /project/<board>/apps/<project>/src/apps/events
- Utils
  /project/<board>/apps/<project>/src/apps/utils

2 Module Introduction

2.1 Event Senders

All the events sent to the APPs layer would be sent via the event sender. Event senders include two modules: BT event, and key event. These two modules are responsible for sending different events.

2.1.1 BT Event

This module sends three groups of events. The EVENT_GROUP_UI_SHELL_BT_SINK group is frequently used by APPs to get information about BT. The EVENT_GROUP_UI_SHELL_BT_CONN_MANAGER group and the EVENT_GROUP_UI_SHELL_BT group are helpful when the EVENT_GROUP_UI_SHELL_BT_SINK group cannot provide enough information.

- EVENT_GROUP_UI_SHELL_BT_SINK group events – The event IDs of the group are defined in sink module in middleware. The BT state change and profile connection change are notified by these events.
- EVENT_GROUP_UI_SHELL_BT_CONN_MANAGER group events – The event IDs of the group are defined in BT connection manager module in middleware.
- EVENT_GROUP_UI_SHELL_BT group events – The event IDs of the group are defined in Bluetooth module in middleware. The event sender callback is registered where the APP needs the BT messages.

2.1.2 Key Event

This module sends EVENT_GROUP_UI_SHELL_KEY group events. This group is used for defining keys. For example, the ID of the power on key is KEY_POWER_ON. Please refer to /project/<board>/apps/<project>/inc/apps/config/apps_config_event_list.h.

2.2 APPs

APPs implement the actions of the application layer and are organized by features and functions. Each APP is composed of one or more activities. The activities are managed by UI shell. Every activity receives events of the EVENT_GROUP_UI_SHELL_SYSTEM group to follow the management to create, destroy, resume, pause, refresh and result.

The following sections introduce the basic APPs in MediaTek IoT SDK. You can base on the design to add new APPs.

2.2.1 Home Screen APP

Home Screen APP works as settings. The app_home_screen_idle_activity is the only activity of Home Screen APP, which is the first priority when the system is in idle state.

- Home Screen APP processes the common key event requests, such as power_on, power_off, BT_discoverable, pairing, reconnect last device and reset paired devices.

The app_home_screen_idle_activity contains a BT connection component, which helps Home Screen APP to process BT and BT sink events, and changes the connection state of Home Screen APP. The relationship between BT sink state and connection state is shown in Table 1.

Table 1 Definitions of the BT State in the Home Screen APP

BT sink state	Profile connected	BT visible	BT connectable	Connection state
NONE	N/A	N/A	N/A	BT OFF
POWER_ON	N/A	TRUE	TRUE	Discoverable
POWER_ON	N/A	FALSE	TRUE	Connectable
POWER_ON	N/A	FALSE	FALSE	Non-discoverable Non-connectable
>= CONNECTED	FALSE	FALSE	FALSE	Non-discoverable Non-connectable
>= CONNECTED	TRUE	FALSE	FALSE	Connected to SP

2.2.2 Music APP

Music APP controls the playback of music. This APP is composed of two activities: app_music_idle_activity and app_music_activity. The idle activity is created when the device is powered on, and the music activity will be started while streaming. App_music_idle_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY and EVENT_GROUP_UI_SHELL_BT_SINK.

- EVENT_GROUP_UI_SHELL_KEY
  - The key event handled by app_music_idle_activity is shown in Table 2. States and keys are

handled by app_music.

- EVENT_GROUP_UI_SHELL_BT_SINK
  - When a BT_SINK_SRV_EVENT_STATE_CHANGE event is received, the APP checks whether the state is changed to streaming and start app_music_activity.

App_music_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY, EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_APP_INTERACTION.

- EVENT_GROUP_UI_SHELL_KEY
  - The key event handled by app_music_activity is shown in Table 2. States and keys are handled by app_music.
- EVENT_GROUP_UI_SHELL_BT_SINK
  - When a BT_SINK_SRV_EVENT_STATE_CHANGE event is received, the APP checks whether the streaming is ended and finishes itself.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
  - When an APPS_EVENTS_INTERACTION_UPDATE_MMI_STATE event is received, the APP sets MMI state.

Table 2 States and Keys Handled by app_music

Activity	State	Key handle
Music idle activity	Power on Connected	Play
Music activity	Streaming	Pause Next / Forward Volume Up / Down

2.3 Project Config

Settings are located in /project/<board>/apps/<project>/inc/project_config.h

2.3.1 BT_LE_CONNECTION_MAX

This variable is used for indicating the maximum number of LE connections. This variable would also infect BT_LE_CONNECTION_BUF_SIZE. Both are related to LE connection limitation.

2.3.2 BT_CONNECTION_MAX

This variable is used for indicating the maximum number of BR/EDR connections. This variable would also affect BT_CONNECTION_BUF_SIZE. Both are related to BR/EDR connection limitation.

2.3.3 BT_AVRCP_TOTAL_LINK_NUM

This variable is used for indicating the maximum number of AVRCP connections. This variable would also affect BT_AVRCP_LINK_BUF_SIZE. Both are related to AVRCP connection limitation. If you want to adjust this value, BT Lib would also need to update.

2.3.4 BT_A2DP_TOTAL_LINK_NUM

This variable is used for indicating the maximum number of AVRCP connections. This variable would also affect BT_A2DP_LINK_BUF_SIZE. Both are related to A2DP connection limitation. If you want to adjust this value, BT Lib would also need to update.

2.3.5 BT_A2DP_SEP_TOTAL_NUM

This variable is used for indicating the maximum number of A2DP SEP. This variable would also infect #define BT_A2DP_SEP_BUF_SIZE.

2.4 BT Connection Manager

The Bluetooth connection manager provides functions that are related to connections such as connect, disconnect and scan mode. The APP can use those functions to implement connection related procedures. For details of those functions, please refer to /middleware/MTK/inc/bt_connection_manager.h.

2.4.1 Functions and Events

Some functions communicate with a remote device, so the functions will not return the result immediately and you will need to handle the event.

Table 3 Function and Event of Connection Manager

Function	Corresponding Event	Description
bt_cm_connect	BT_CM_EVENT_REMOTE_INFO_UPDATE	Used for profile connection
bt_cm_disconnect	BT_CM_EVENT_REMOTE_INFO_UPDATE	Used for disconnection
bt_cm_force_disconnect	BT_CM_EVENT_REMOTE_INFO_UPDATE	Used for disconnecting all BR/EDR devices
bt_cm_scan_mode	BT_CM_EVENT_VISIBILITY_STATE_UPDATE	Used for setting scan mode

2.4.1.1 BT_CM_EVENT_REMOTE_INFO_UPDATE

This event is sent when the connection status changes. All the information is included in bt_cm_remote_info_update_ind_t.

```
typedef struct {
    bt_bd_addr_t                address;
    bt_cm_acl_link_state_t      pre_acl_state;
    bt_cm_acl_link_state_t      acl_state;
    bt_cm_profile_service_mask_t pre_connected_service;
    bt_cm_profile_service_mask_t connected_service;
    bt_cm_profile_service_t     update_service;
    bt_status_t                 reason;
} bt_cm_remote_info_update_ind_t;
```

BT_CM_EVENT_REMOTE_INFO_UPDATE is notified in three cases.

- ACL link connect/disconnect
- ACL link encrypted
- Profile connect/disconnect

The ACL link status is recorded in **acl_state** of bt_cm_remote_info_update_ind_t. Below is the corresponding definition.

```
#define BT_CM_ACL_LINK_DISCONNECTED    (0x00)
#define BT_CM_ACL_LINK_DISCONNECTING   (0x01)
#define BT_CM_ACL_LINK_PENDING_CONNECT (0x02)
#define BT_CM_ACL_LINK_CONNECTING      (0x03)
#define BT_CM_ACL_LINK_CONNECTED       (0x04)
#define BT_CM_ACL_LINK_ENCRYPTED       (0x05)
```

The connected profile is recorded in **connected_service** of bt_cm_remote_info_update_ind_t. The bit definition is shown below. For example, if the value of **connected_service** is 0x50, then the current connected profile is A2DP_SINK and AVRCP.

```
typedef enum {
    BT_CM_PROFILE_SERVICE_NONE = 0,
    :
    BT_CM_PROFILE_SERVICE_A2DP_SINK,
    BT_CM_PROFILE_SERVICE_A2DP_SOURCE,
    BT_CM_PROFILE_SERVICE_AVRCP,

    BT_CM_PROFILE_SERVICE_CUSTOMIZED_BEGIN,
    BT_CM_PROFILE_SERVICE_CUSTOMIZED_END,

    BT_CM_PROFILE_SERVICE_MAX
} bt_cm_profile_service_t;
```

The reason of disconnection is recorded in **reason** of bt_cm_remote_info_update_ind_t. For the definition of disconnection reasons, please refer to /prebuild/middleware/MTK/bluetooth/inc/bt_hci.h and /prebuild/middleware/MTK/bluetooth/inc/bt_type.h. For example, if the application sends a disconnection request, the profile connection is disconnected first, and then ACL connection is disconnected. So, you will receive **two BT_CM_EVENT_REMOTE_INFO_UPDATE events** for **one** disconnecting operation.

```
{
    bt_cm_remote_info_update_ind_t *remote_update =
                                        (bt_cm_remote_info_update_ind_t *)data
    if (remote_update->update_service)
        //It means this event is updated for profile connection
        //meaning of reason should check bt_type.h
    else
        //It means this event is updated for ACL connection
        //meaning of reason should check bt_hci.h
}
```

2.4.1.2 BT_CM_EVENT_VISIBILITY_STATE_UPDATE

This event notifies that the status of scan mode changes, which happens after bt_cm_scan_mode is called. The definition of the data structure is as below.

```
typedef struct {
    bool visibility_state;
    bool connectivity_state;
} bt_cm_visibility_state_update_ind_t;
```

- Visibility_state means whether DUT can be found by a remote device or not.
- Connectivity_state means whether DUT can be connected to a remote device or not.

2.5 Sink Service

Sink service contains A2DP SINK and AVRCP profile.

- A2DP SINK is used for receiving media data.
- AVRCP is used for music operations.

2.5.1 Action and Event

The interface of sink service is Action and Event. Below is a list of the actions and events.

Table 4 Actions and Events of Sink Service

Action	Corresponding Event	Description
BT_SINK_SRV_ACTION_PLAY	BT_SINK_SRV_EVENT_AVRCP_STATUS_CHANGE	Play music
BT_SINK_SRV_ACTION_PAUSE	BT_SINK_SRV_EVENT_AVRCP_STATUS_CHANGE	Pause music
BT_SINK_SRV_ACTION_NEXT_TRACK	N/A	Next track
BT_SINK_SRV_ACTION_PREV_TRACK	N/A	Previous Track
BT_SINK_SRV_ACTION_PLAY_PAUSE	BT_SINK_SRV_EVENT_AVRCP_STATUS_CHANGE	Toggle play/pause state
BT_SINK_SRV_ACTION_FAST_FORWARD	N/A	Fast Forward
BT_SINK_SRV_ACTION_REWIND	N/A	Rewind
BT_SINK_SRV_ACTION_GET_PLAY_STATUS	BT_SINK_SRV_EVENT_AVRCP_GET_PLAY_STATUS_CNF	Complete the event of get play status
BT_SINK_SRV_ACTION_GET_CAPABILITY	BT_SINK_SRV_EVENT_AVRCP_GET_CAPABILITY_CNF	Complete the event of get capability
BT_SINK_SRV_ACTION_GET_ELEMENT_ATTRIBUTE	BT_SINK_SRV_EVENT_AVRCP_GET_ELEMENT_ATTRIBUTES_CNF	Complete the event of get element attribute
N/A	BT_SINK_SRV_EVENT_STATE_CHANGE	Event that notifies that the sink service has changed

2.5.1.1 Action

Action is a function supported by the sink service. The application can send an action request to the sink service by using `bt_sink_srv_send_action`. For example, if you want to play music, you can send the action BT_SINK_SRV_ACTION_PLAY to the sink service.

```
{
    bt_sink_srv_action_t sink_action = BT_SINK_SRV_ACTION_NONE;
    bt_status_t bt_status;

    sink_action = BT_SINK_SRV_ACTION_PLAY;
    bt_status = bt_sink_srv_send_action(sink_action, op);
    if (bt_status != BT_STATUS_SUCCESS)
        // print error message
}
```

2.5.1.2 Event

2.5.1.2.1 BT_SINK_SRV_EVENT_AVRCP_STATUS_CHANGE

This event is used for indicating AVRCP play status. Every time you send a play or pause command, you receive this event sent from the sink service.

This event is one of the events that can register with a remote device; the sink service has automatically registered this event. You can use the action: `BT_SINK_SRV_ACTION_GET_CAPABILITY` to check other events supported by a remote device. Then, you can use `bt_avrcp_register_notification` provided by the MT793X to register an event.

```
typedef union {
    bt_sink_srv_state_change_t                  state_change;
    bt_sink_srv_event_avrcp_status_changed_ind_t avrcp_status_change;
} bt_sink_srv_event_param_t;
```

```
typedef struct {
    bt_bd_addr_t   address;
    bt_avrcp_status_t    avrcp_status;
} bt_sink_srv_event_avrcp_status_changed_ind_t;
```

```
#define    BT_AVRCP_STATUS_PLAY_STOPPED  0x00
#define    BT_AVRCP_STATUS_PLAY_PLAYING  0x01
#define    BT_AVRCP_STATUS_PLAY_PAUSED   0x02
#define    BT_AVRCP_STATUS_PLAY_FWD_SEEK 0x03
#define    BT_AVRCP_STATUS_PLAY_REV_SEEK 0x04
#define    BT_AVRCP_STATUS_PLAY_ERROR    0xFF
```

2.5.1.2.2 BT_SINK_SRV_EVENT_AVRCP_GET_PLAY_STATUS_CNF

This event is the complete response of the action: BT_SINK_SRV_ACTION_GET_PLAY_STATUS. This event contains data in the structure below.

```
typedef struct {
    bt_status_t status;
    bt_bd_addr_t address;
    uint32_t song_length;
    uint32_t song_position;
    bt_avrcp_media_play_status_event_t play_status;
} bt_sink_srv_avrcp_get_play_status_cnf_t;
```

```
#define    BT_AVRCP_EVENT_MEDIA_PLAY_STOPPED 0x00
#define    BT_AVRCP_EVENT_MEDIA_PLAYING      0x01
#define    BT_AVRCP_EVENT_MEDIA_PAUSED       0x02
#define    BT_AVRCP_EVENT_MEDIA_FWD_SEEK     0x03
#define    BT_AVRCP_EVENT_MEDIA_REV_SEEK     0x04
#define    BT_AVRCP_EVENT_MEDIA_ERROR        0xFF
```

When you receive this event, you can check the value of the status. If the status is BT_STATUS_SUCCESS, then you can continue to read the remaining elements.

2.5.1.2.3 BT_SINK_SRV_EVENT_AVRCP_GET_CAPABILITY_CNF

This event is the complete response of the action: BT_SINK_SRV_ACTION_GET_CAPABILITY.

```
typedef struct {
    bt_status_t status;
    bt_bd_addr_t address;
    uint8_t number;
    uint8_t capability_value[BT_SINK_SRV_MAX_CAPABILITY_VALUE_LENGTH];
} bt_sink_srv_avrcp_get_capability_cnf_t;
```

When you receive this event, you can check the value of the status. If the status is BT_STATUS_SUCCESS, then you can continue to read the remaining elements. The element: `number` indicates the number of effective values in `capability_value`. For example, if the value of `number` is 3, then you should read `capability_value[0]`, `capability_value[1]` and `capability_value[2]`.

2.5.1.2.4 BT_SINK_SRV_EVENT_AVRCP_GET_ELEMENT_ATTRIBUTES_CNF

This event is the complete response of the action: BT_SINK_SRV_ACTION_GET_ELEMENT_ATTRIBUTE.

```
typedef struct {
    bt_status_t status;
    bt_bd_addr_t address;
    bt_avrcp_metadata_packet_type_t packet_type;
    uint16_t length;
    union {
        struct {
            uint8_t number;
            bt_avrcp_get_element_attributes_response_value_t *attribute_list;
        };
        uint8_t *data;
    };
}bt_sink_srv_avrcp_get_element_attributes_cnf_t;
```

Below is the definition of bt_avrcp_metadata_packet_type_t.

```
#define BT_AVRCP_METADATA_PACKET_TYPE_NON_FRAGMENT 0x00
#define BT_AVRCP_METADATA_PACKET_TYPE_START      0x01
#define BT_AVRCP_METADATA_PACKET_TYPE_CONTINUE   0x02
#define BT_AVRCP_METADATA_PACKET_TYPE_END        0x03
```

```
typedef struct {
    bt_avrcp_media_attribute_t attribute_id;
    uint16_t character_set_id;
    uint16_t attribute_value_length;
    uint8_t attribute_value[1];
}bt_avrcp_get_element_attributes_response_value_t;
```

Below is the definition of bt_avrcp_media_attribute_t.

```
#define BT_AVRCP_MEDIA_ATTRIBUTE_TITLE              0x01
#define BT_AVRCP_MEDIA_ATTRIBUTE_ARTIST_NAME        0x02
#define BT_AVRCP_MEDIA_ATTRIBUTE_ALBUM_NAME         0x03
#define BT_AVRCP_MEDIA_ATTRIBUTE_MEDIA_NUMBER       0x04
#define BT_AVRCP_MEDIA_ATTRIBUTE_TOTAL_MEDIA_NUMBER 0x05
#define BT_AVRCP_MEDIA_ATTRIBUTE_GENRE              0x06
#define BT_AVRCP_MEDIA_ATTRIBUTE_PLAYING_TIME       0x07
```

When you receive this event, you can check the value of the status. If the status is BT_STATUS_SUCCESS, then you can continue to read the remaining elements.

- The element packet_type indicates whether the packet is fragmented or not. You should wait until the packet_type becomes BT_AVRCP_METADATA_PACKET_TYPE_END and then start to parse this event.
- The element `number` indicates the length of `attribute_list`. For example, if the number is 2, you should handle `attribute_list`[0] and `attribute_list[1]` .

2.5.1.2.5 BT_SINK_SRV_EVENT_STATE_CHANGE

This event indicates the state of the sink service.

```
typedef struct {
    bt_sink_srv_state_t previous;
    bt_sink_srv_state_t current;
} bt_sink_srv_state_change_t;
```

Below is the definition of `bt_sink_srv_state_t` .

```
#define BT_SINK_SRV_STATE_NONE              (0x0000)
#define BT_SINK_SRV_STATE_POWER_ON          (0x0001)
#define BT_SINK_SRV_STATE_CONNECTED         (0x0002)
#define BT_SINK_SRV_STATE_STREAMING         (0x0004)
```

2.5.2 Streaming Data

As the role of A2DP sink, streaming data are sent from a remote device. Below is the flow of media data.

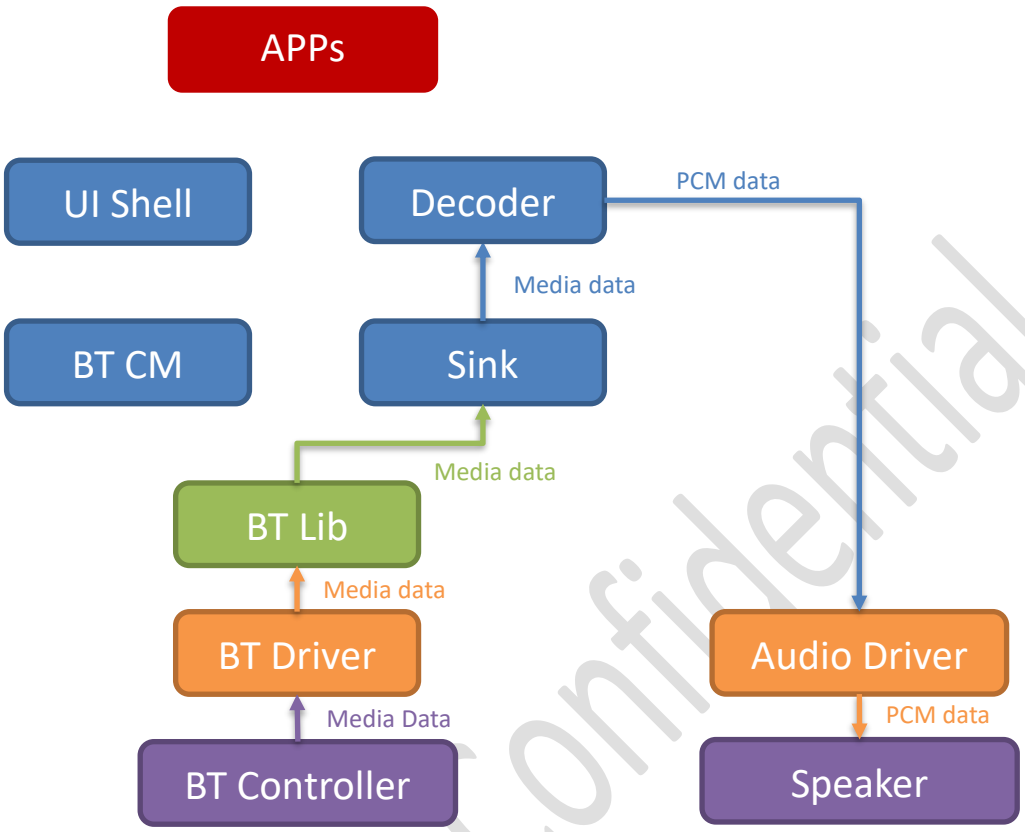

Figure 3 Flow of Media Data

BT Lib sends BT_A2DP_STREAMING_RECEIVED_IND to the sink service, and then the sink service sends the media data to the decoder. After the decoder finishes the decode procedure, the PCM data are sent to the audio driver.

To avoid sound stuttering, the sink service puts some media data into a queue and then triggers the decoder to decode the data. After music is stopped, all the data in the queue is freed.

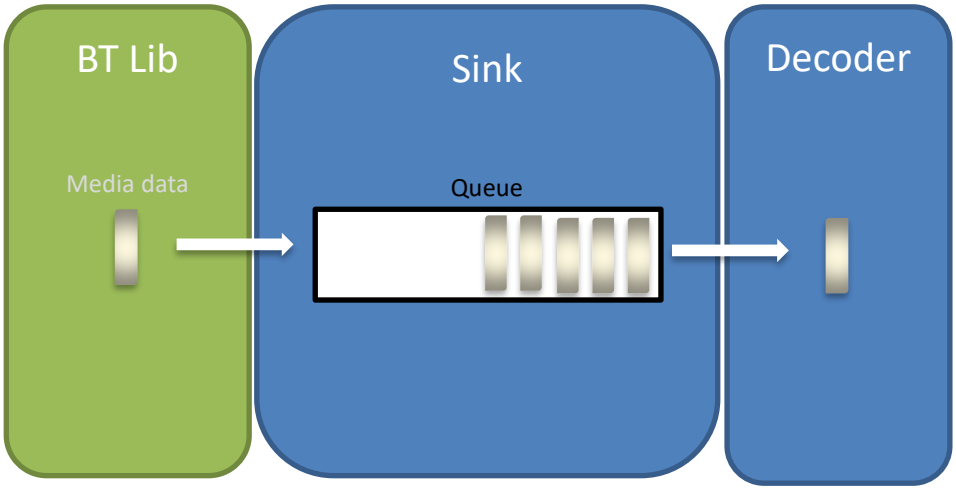

Figure 4 Media Data Queue Mechanism

2.6 Device Manager

The device manager is used for managing device information. The number of paired device is configured in /middleware/MTK/Bluetooth_service/inc/bt_device_manager_config.h.

```
#define BT_DEVICE_MANAGER_MAX_PAIR_NUM        3
```

2.7 Examples

The section gives some examples to show how the SDK works.

2.7.1 Profile Connect

For the connection procedure, the application should send a command by using the function provided by the BT connection manager, and then the BT connection manager would send an action for the sink service, and then the sink service sends the command to the lower layer.

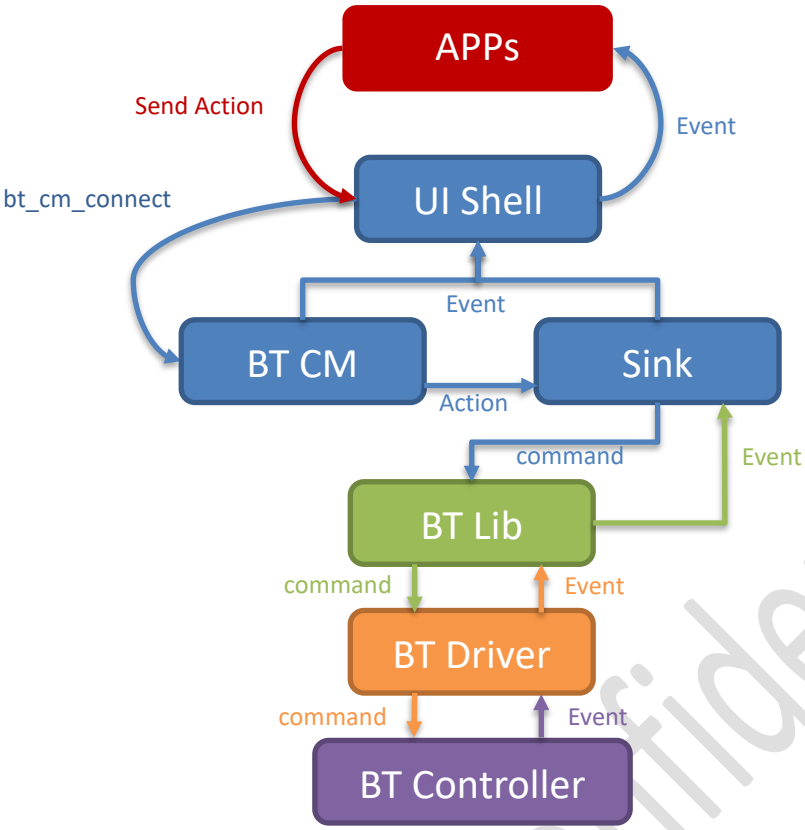

Figure 5 Example Flow of Connection

2.7.2 A2DP Music Playing

For music playing, the application can send an action to the sink service directly.

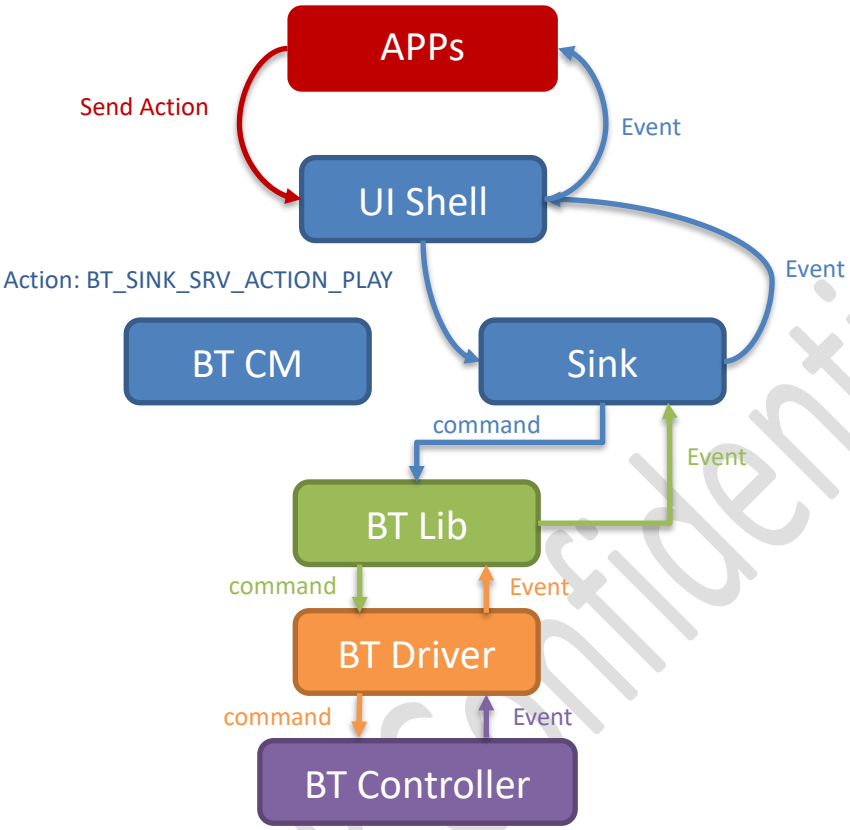

Figure 6 Example Flow of Music Playing

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.