Airoha IoT SDK Ultra Low Latency Developer's Guide

Version: 1.5

Release date: 28 December 2021

Document Revision History

Revision	Date	Description
1.0	4 March 2021	• Initial release
1.1	27 April 2021	• Added the Chapter 3. Developing the ULL UI
1.2	10 May 2021	• Added the description of reconnection in ULL UI • Added the ULL state machine
1.3	23 June 2021	• Added ULL critical data transmit-receive • Added ULL user data transmit-receive
1.4	28 July 2021	• Added media key actions • Added PC tool control call of smartphone • Added automatically reduce the volume of game audio when chat audio is present
1.5	28 December 2021	• Added gaming mode

Table of Contents

Lists of Tables and Figures

1. Introduction

Ultra-Low Latency (ULL) is an Airoha proprietary technology to support less than 20ms downlink voice/audio latency for headsets/earbuds over Bluetooth with a well-matched Bluetooth-Dongle.

There are two roles in the ULL profile:

- **ULL_Server** — a device that usually has a capability with USB Audio Sound Card and relay the audio data to remote device via wireless communication. It provides functions as below:
  - 2-RX (PC→Device) & 1-TX (Device→PC)
  - 2-RX audio mixing ratio control
  - Firmware update via USB
  - Support USB HID
  - Play/stop state notification
- **ULL_Client** — a device that acts as **ULL_Server**'s remote audio input and output. There are two kinds of common device: Headset and Earbuds;

This document guides you through:

- Support for Bluetooth with the library description and supported reference examples.
- Detailed descriptions of the ULL profiles.
- Custom application development and debugging logs.

1.1. Profile Overview

Figure 1 shows the protocols and entities used in this profile.

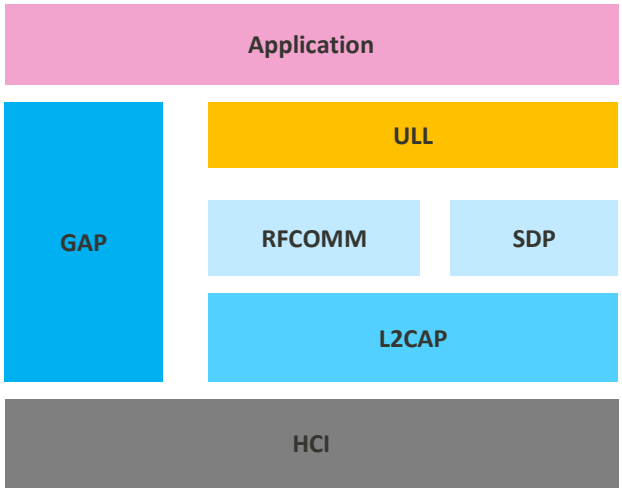

Figure 1. Protocol Model

The HCI, L2CAP, GAP, RFCOMM, and SDP protocols are described in the *Airoha_IoT_SDK_Bluetooth_Developers_Guide.pdf* document under the <SDK_root>/mcu/doc folder.

1.2. Usage Scenario

ULL_Server is a device that supports USB Audio Sound Card capability. It encodes USB PCM streaming to OPUS format and transmits to **ULL_Client** via Bluetooth technology.

ULL_Client supports multi-link connection (ULL_Server + Smartphone's HFP).

The ULL usage scenario and multi-link are shown in Figure 2.

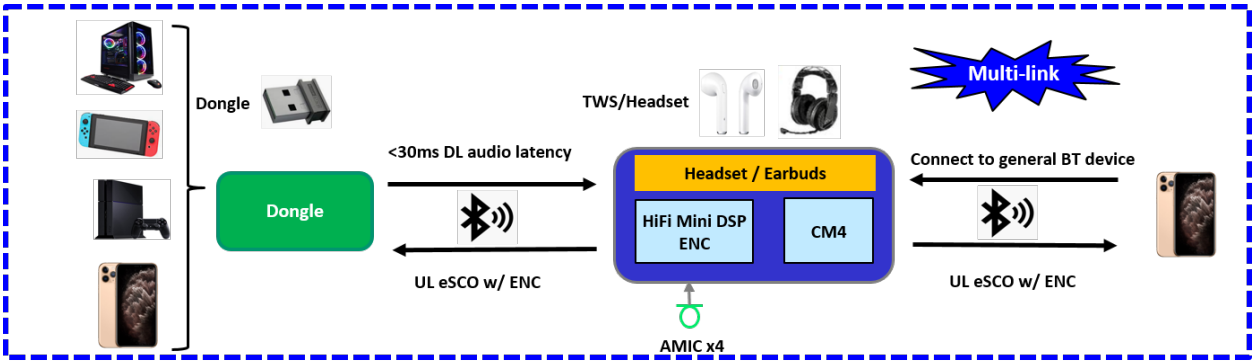

Figure 2. ULL usage scenario

1.3. Related SDK Library Requested

ULL can only be run on Airoha IoT SDK for BT-Audio platform with the requested library files to interface the Bluetooth with C source and header files related to the platform, as shown in Table 1.

Table 1. Airoha IoT SDK library support for ULL

Module	File Name	Location	Function
Bluetooth	libbt.a	/middleware/bluetooth/lib/	BR/EDR and Bluetooth LE stack library
	libbtdriver_[chip].a		Bluetooth driver library
	libbt_spp.a		SPP library
	libbt_aws_mce.a		MCSync library, including MCSync implementation
	bt_platform.h	/middleware/bluetooth/inc/	Interface for Bluetooth tasks
	bt_type.h		Common data types
	bt_system.h		Interface for the system, such as power on or off, memory initiation and callback APIs for event handling
	bt_uuid.h		Interface for the UUID
	bt_codec.h		Interface for the codec
	bt_spp.h		Interface for the SPP
	bt_aws_mce.h		Interface for the MCSync
	bt_gap.h		Interface for the GAP
	bt_sdp.h		Interface for the SDP

Module	File Name	Location	Function
	bt_os_layer_api.h		Wrapper APIs for RTOS, memory, advanced encryption standard (AES) and rand
	bt_debug.h		Encapsulated debugging interface
	bt_hci_log.h		Encapsulated interface for the HCI logging
	bt_os_layer_api.c	/middleware/bluetooth/src/	Encapsulated interface for system, memory or AES. Developers can replace the implementation when porting to other platforms
	bt_debug.c		Encapsulated debugging interface. Developers can replace the implementation when porting to other platforms
	bt_hci.c		Encapsulated interface for the HCI logging. Developers can replace the implementation when porting to other platforms
	bt_task.c		The default Bluetooth task entry function
	bt_log.c		The definition for the debugging string used in BT Stack library

2. The ULL Profile

2.1. The ULL Message Sequence

The ULL procedure can be established using the message sequence. The message sequence for each process is described below:

- Air pairing
- Connection establishment
- Connection release
- Set 2-RX mixing ratio
- Set downlink latency
- Critical data transmit-receive
- User data transmit-receive

2.1.1. Air Pairing

The Air Pairing procedure is used for Client and Service become a couple. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

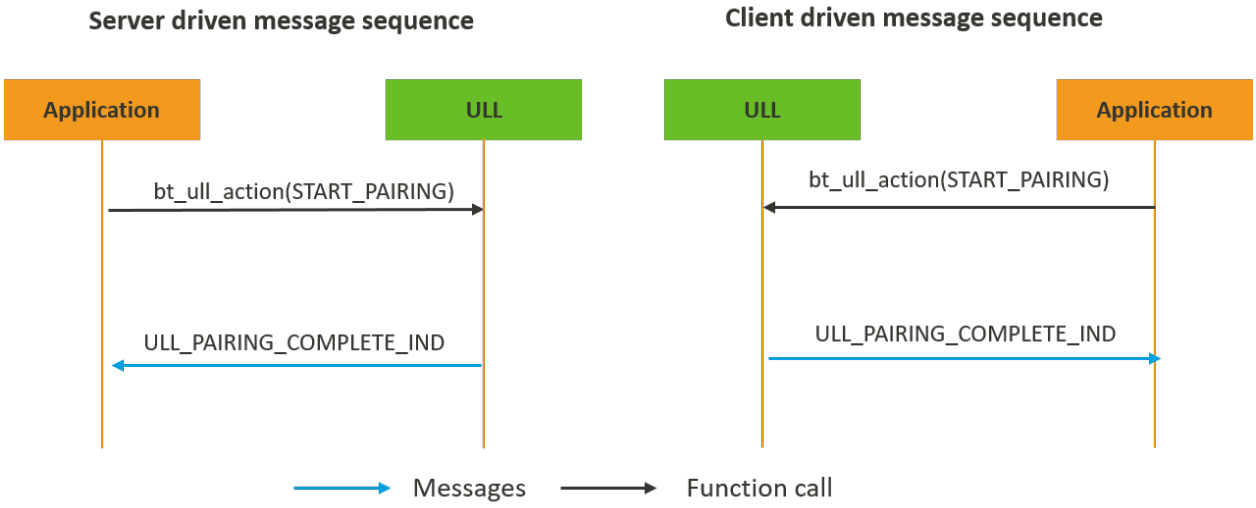

Figure 3. ULL Server and Client pairing

2.1.2. Connection Establishment

Use the connection establishment operation to establish a connection between Server and Client. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

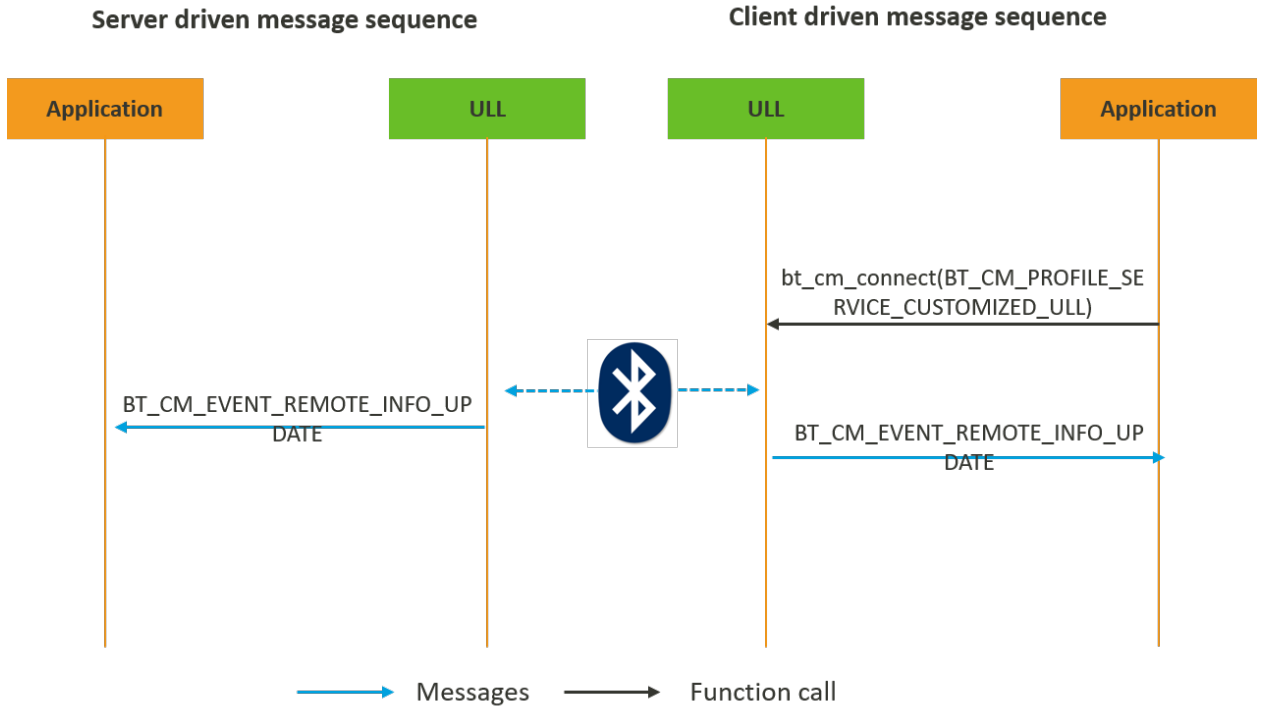

Figure 4. ULL connection establishment message sequence

2.1.3. Connection Release

The connection release procedure disconnects the ULL profile. Both Server and Client can initiate the disconnection procedure. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

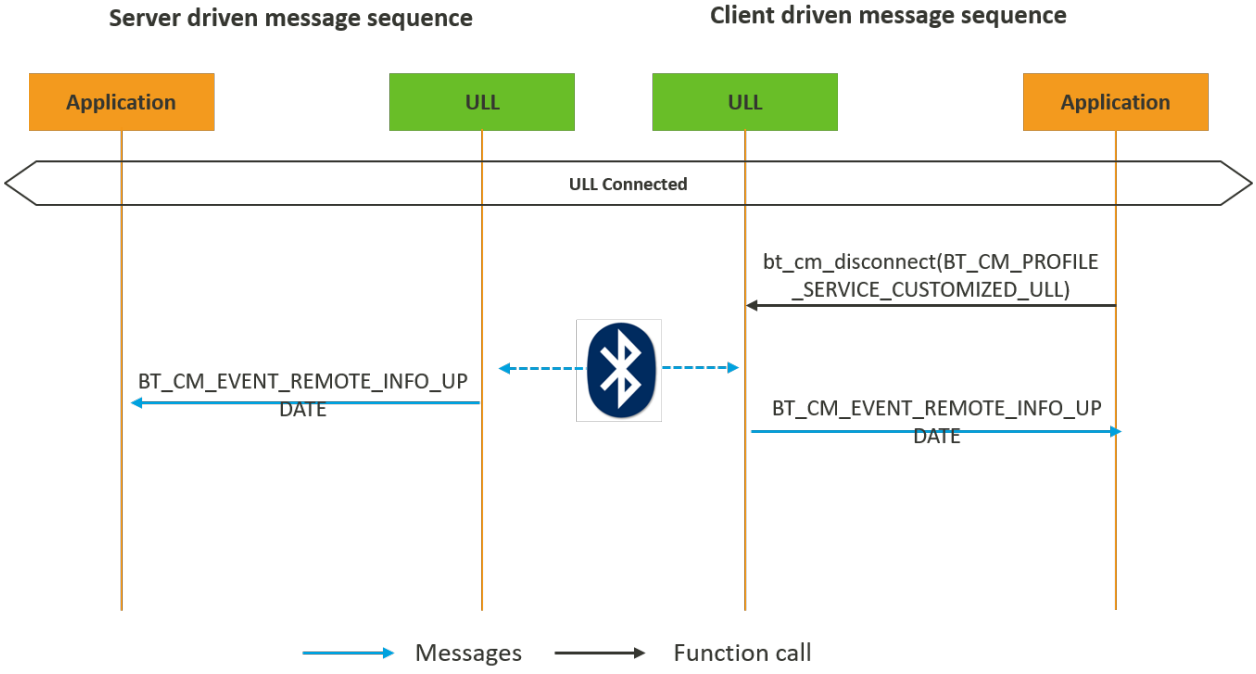

Figure 5. Disconnect ULL profile

2.1.4. Set 2-RX Mixing Ratio

Use the set mixing ratio operation to control USB Host (Ex. PC) so that you can combine two audio streams together. It supports 0% ~ 100% ratio adjustment for two audio streams independently. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

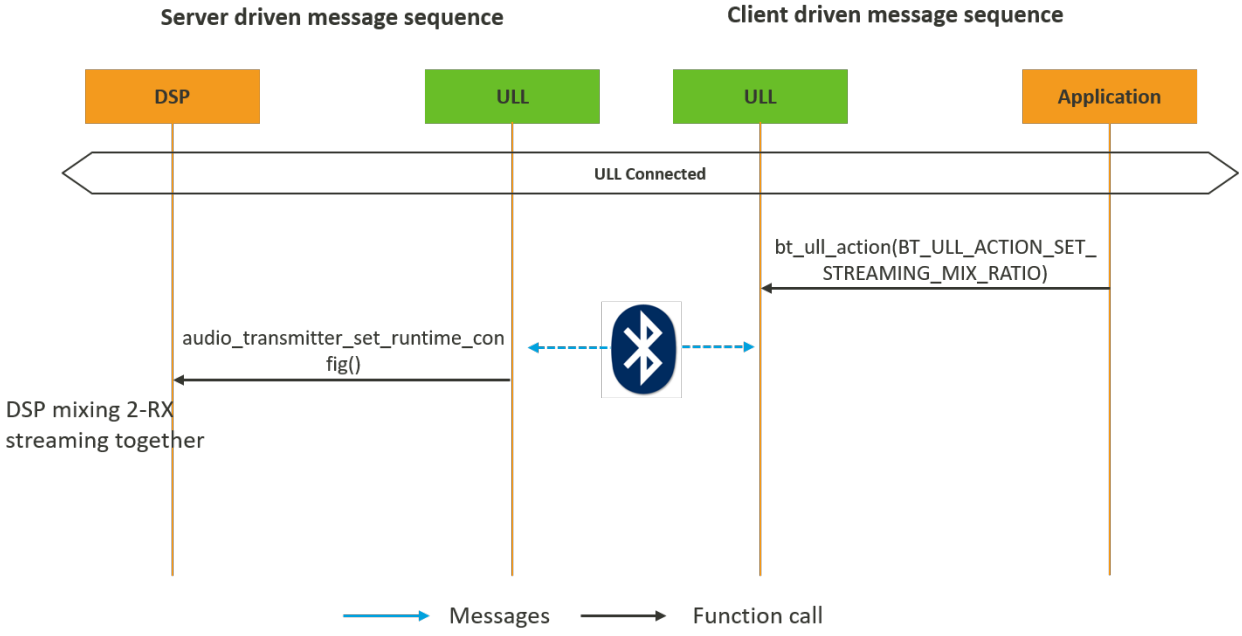

Figure 6. Set 2-RX mixing ratio on Client

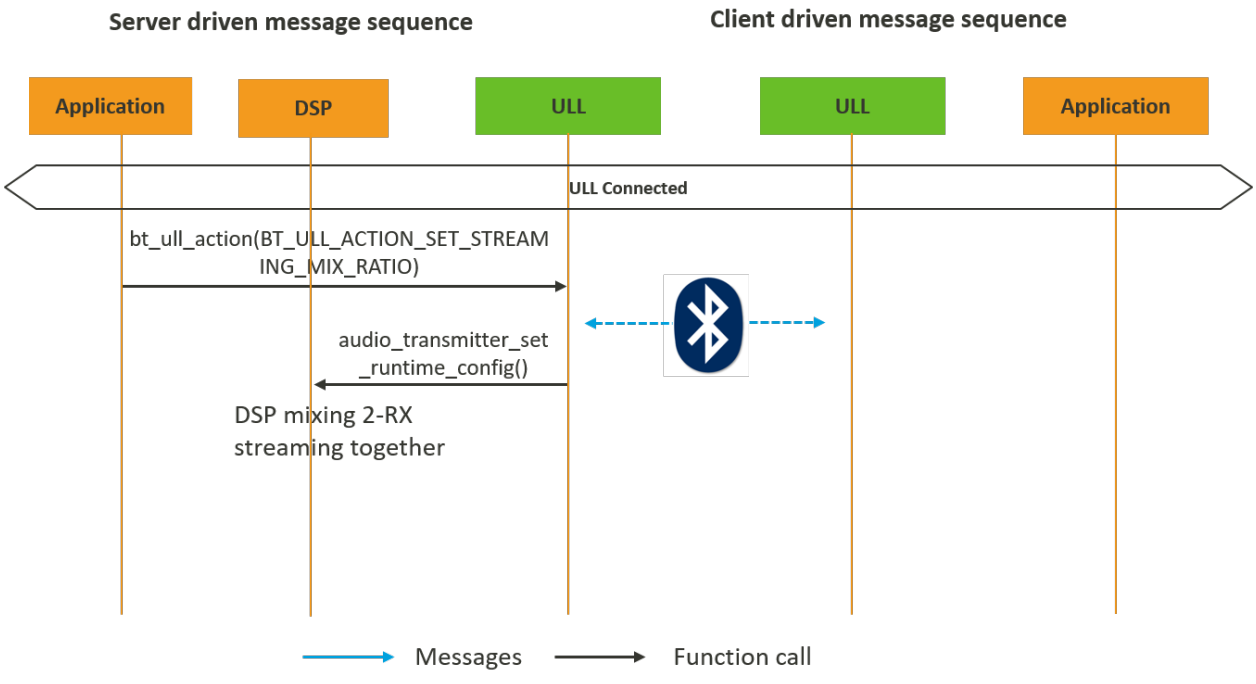

Figure 7. Set 2-RX mixing ratio on Server

2.1.5. Set Downlink Latency

Due to Bluetooth bandwidth limitation, sometimes we should change the ULL downlink streaming latency to a different value (Ex. 60ms) on ULL_Client. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

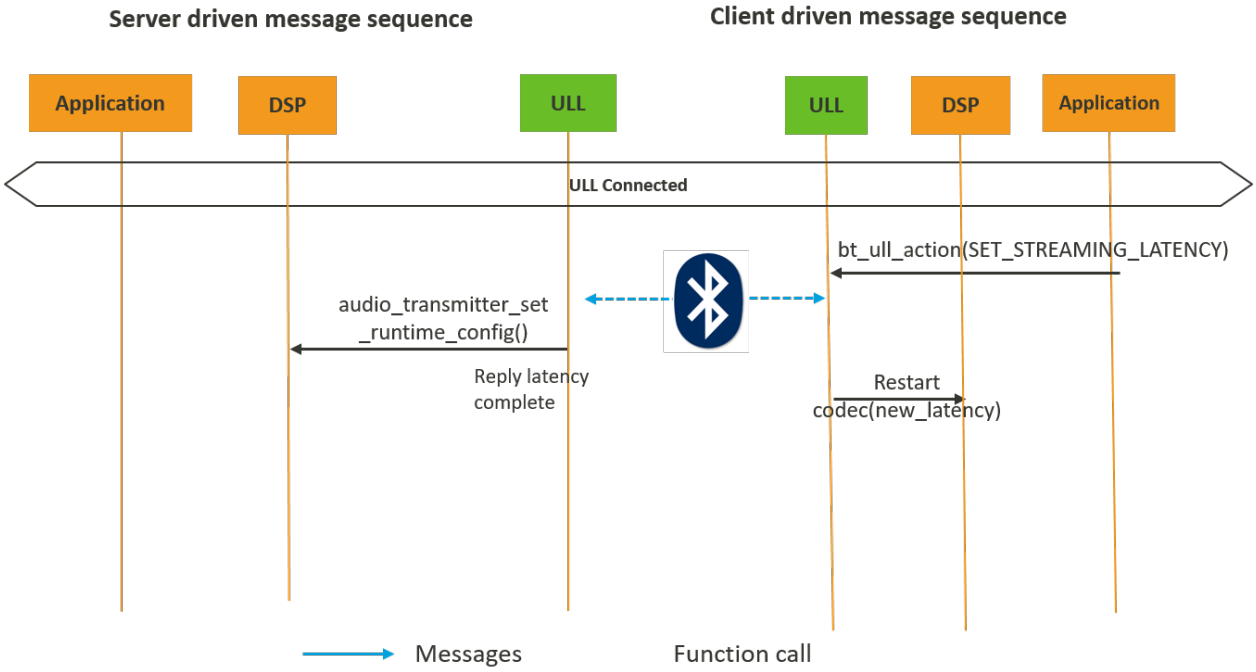

Figure 8. Set downlink latency

2.1.6. Critical data transmit-receive

Use the critical data transmit-receive to exchange some unreliably continuous data (such as sensor data) with a flush timeout between Server and Client. The maximum length of critical data is 100 bytes. There is currently only support for Client to send critical data to Server. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

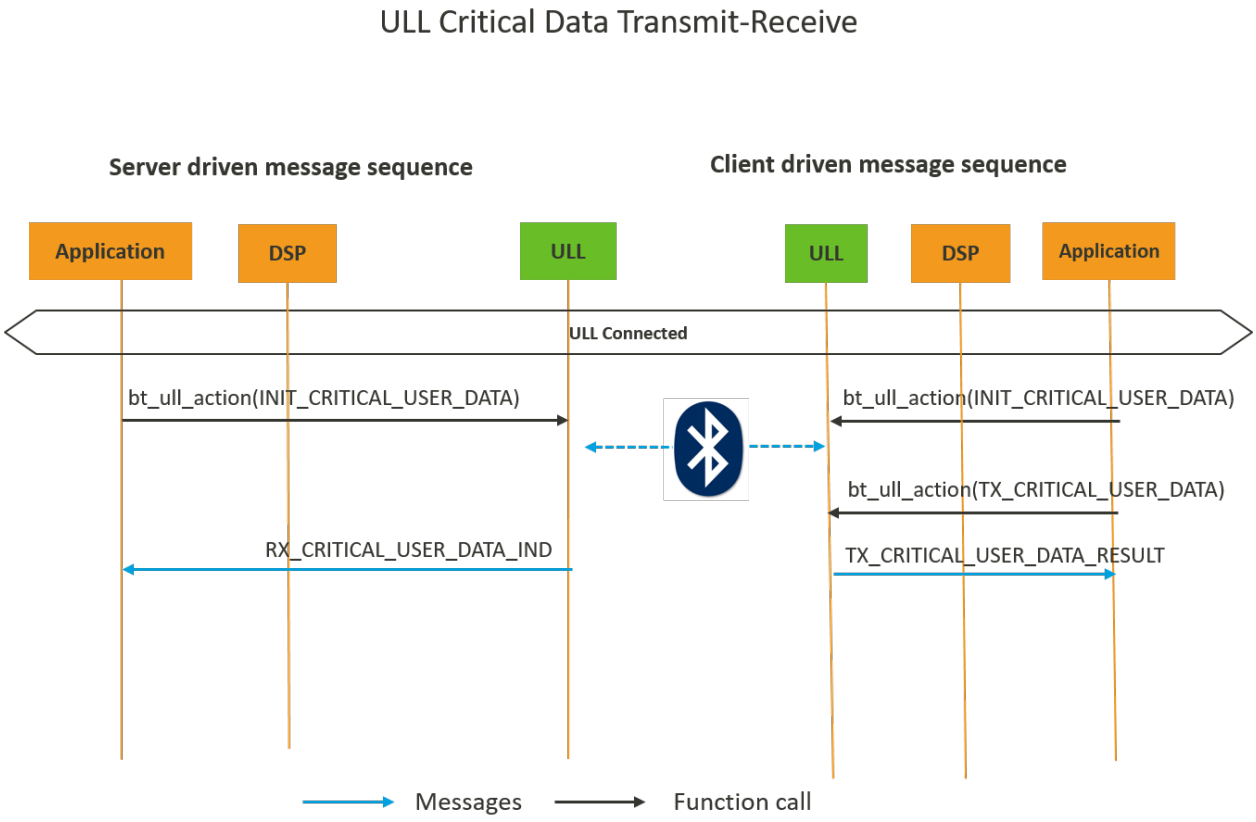

Figure 9. Critical data transmit-receive

2.1.7. User data transmit-receive

Use the user data transmit-receive to exchange user defined data between Server and Client. For more details, refer to <SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h.

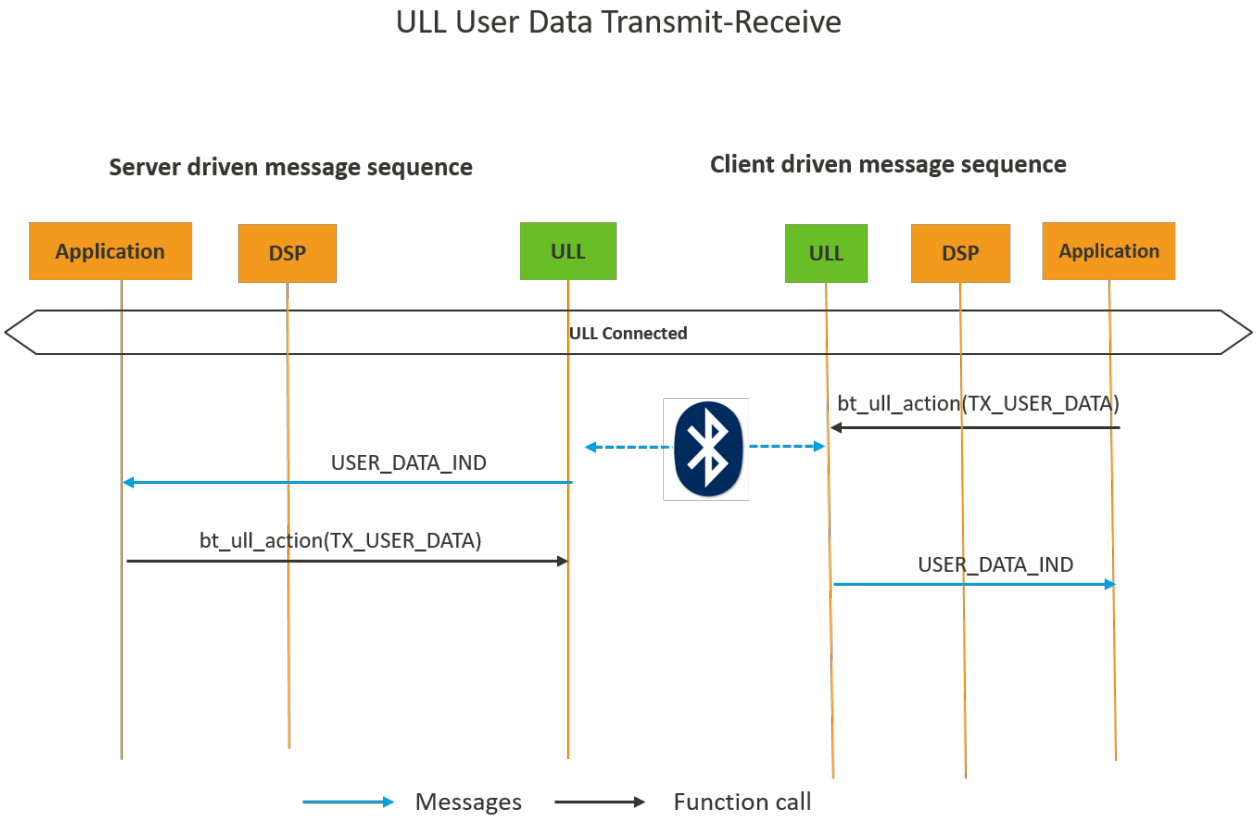

Figure 10. User data transmit-receive

2.2. Using the ULL APIs

This section describes how to use the ULL APIs for application development. The functionality of the ULL APIs is implemented in the module bt_ultra_low_latency, related APIs can be found in `<SDK_root>/mcu/middleware/MTK/bt_ultra_low_latency/inc/bt_ull_service.h`, the other header files are used internally, and application cannot use them at any time.

1) Call the `bt_ull_init()` function to start the ULL role during the initiation process in Dongle as Server or Headset/Earbuds as Client when the system powers on.

```
bt_ull_init(role, callback);
```

2) Client call `bt_ull_action()` function to control. Ex. Client increases the volume

```
  bt_ull_volume_t volume_param;
  volume_param.streaming.streaming_interface =
BT_ULL_STREAMING_INTERFACE_SPEAKER;
  volume_param.streaming.port = 0;
  volume_param.action = BT_ULL_VOLUME_ACTION_SET_UP;
  volume_param.channel =  BT_ULL_AUDIO_CHANNEL_DUAL;
  volume_param.volume = 1;
  bt_ull_action(BT_ULL_ACTION_SET_STREAMING_VOLUME,&volume_param,sizeof(volume_pa
ram));
```

3. Developing the ULL UI

3.1. Multi-link Mode and Single Link Mode

3.1.1. Multi-link Mode

Multi-link mode means the DUT can be connected with one dongle and one smartphone at the same time.

There is a feature that provides the “AIR_BT_ULTRA_LOW_LATENCY_A2DP_STANDBY_ENABLE” option to configure if DUT supports the A2DP profile when both the dongle and smartphone are connected with the DUT. In the default setting, when both SRC are connected, the supported profiles and the latency value between the dongle and DUT is described in the table.

Table 2. Configuration of AIR_BT_ULTRA_LOW_LATENCY_A2DP_STANDBY_ENABLE

Feature option enabled	Supported profiles by smartphone	Latency value between DUT and dongle
y	A2DP and HFP	60ms
n	HFP	25ms

If the DUT is connected to only one source device, it still enables the page scan so another device can connect to the DUT. Page scan uses the BT resource, so the latency of the connection between the dongle and the DUT is 60ms.

If Bluetooth powers on in this mode, the device always reconnects to dongle first. If it page failed, the device reconnects to smartphone and dongle alternately.

3.1.2. Single Link Mode

The single link mode means the DUT can be connected to only one source device at a time. The latency of the connection between the dongle and the DUT is 20ms. In this mode, user can use a key to switch the connection between the dongle and the smartphone.

If Bluetooth powers on in this mode, the device reconnects to the last connected device. If page failed, it then reconnects to the smartphone and dongle alternately.

3.1.3. Reconnection Rule

When the device has connected to a SRC device, reconnecting to another SRC is not 100% successful because the Bluetooth bandwidth is limited. For consistency of UI, devices do not reconnect to the second SRC when the first SRC is connected.

3.1.4. Gaming mode

The gaming mode UI is a design which is enabled by setting the feature option AIR_APP_ULL_GAMING_MODE_UI_ENABLE to y. After enabling the gaming mode feature, the headset must be in multi-link mode and user cannot use key to switch the link mode.

If dongle is connected, user can use key to switch the gaming mode to on or off status. When the headset is in gaming mode on status, the headset should make sure the latency is as low as possible. When the headset is in gaming mode off status, the headset should make the smartphone connect and connect the A2DP profile. The details for each action are shown in the table below.

Table 3. Action when gaming mode switch

Gaming mode	Status	Action	Retry count(latency)
OFF->ON	SP connected	Disable A2DP	4 (25ms)
OFF->ON	SP disconnected	Disable page scan	4 (25ms)
ON->OFF	SP connected	Reconnect A2DP	4 (40ms)
ON->OFF	SP disconnected	Enable page scan	4 (60ms)

If dongle is not connected, press key to switch gaming mode will be failed and headset will play a VP to notify user.
The default key usage in demo project is double-long press.

3.2. Wired USB Audio and Aux In

This feature is only supported on headset projects.

When the wired USB audio is enabled or Aux in is plugged in, the DUT disconnects the dongle connection. If the DUT currently connected to a smartphone, it will try to reconnect the A2DP profile.
When wired USB audio is disabled or Aux in is not plugged in, the DUT tries to reconnect the dongle.

3.3. State machine diagram

The state machine diagram includes connection, disconnection, aux or USB audio in or out, and using the key switch connection.

The multi-link mode has one more state than single link mode: Connected 2 SRC which is shown in blue in this diagram.

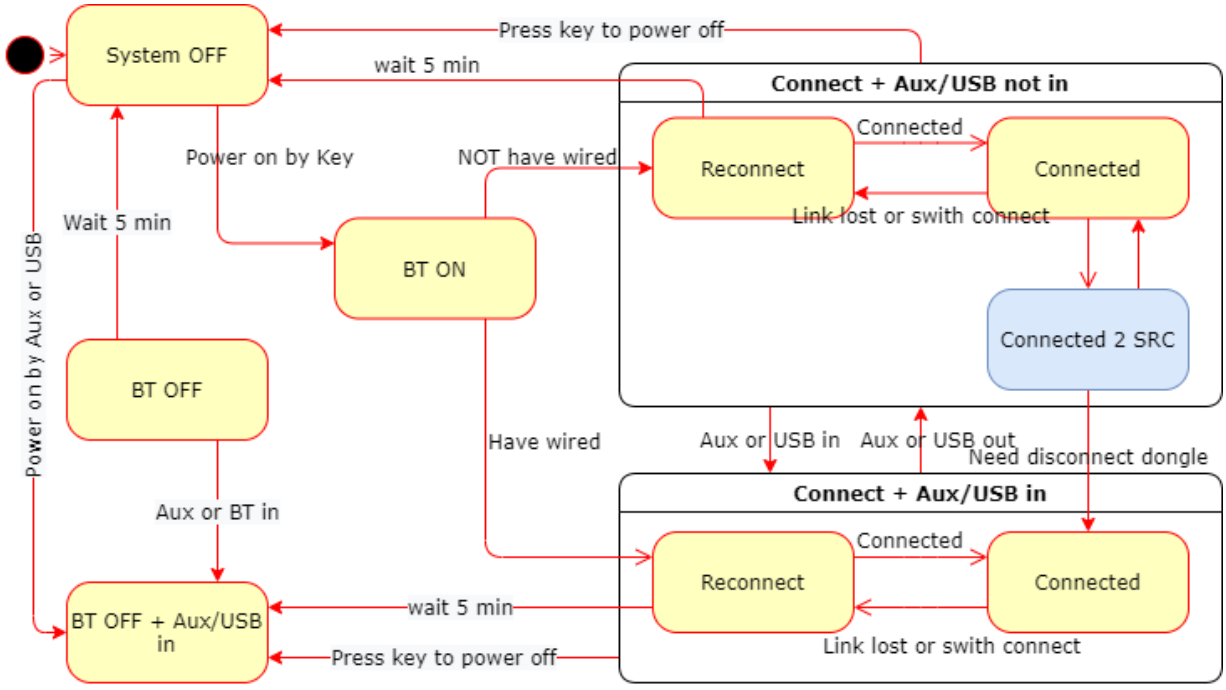

Figure 11. ULL state machine

3.4. ULL Profile Event

3.4.1. Send Event of ULL

The ULL profile notifies new events in the function bt_ulla_callback(). The code is ready in bt_ulla_callback() in <project>\src\apps\events\apps_events_bt_event.c.

In the function bt_ulla_callback(), it sends the events to the UI task. The event group is EVENT_GROUP_BT_ULTRA_LOW_LATENCY.

3.4.2. Use the Event

On the dongle, earbuds or headset side, use BT_ULL_EVENT_PAIRING_COMPLETE_IND to get the pairing result. The event is normally used to store the BT address.

On the dongle side, use BT_ULL_EVENT_USB_PLAYING_IND, BT_ULL_EVENT_USB_STOP_IND, ..., to process the stream event from USB. The processing is normally fixed in the SDK code; The customer does not need to make any changes.

3.5. Key Actions

3.5.1. ULL Key Actions

There are some key action that are specifically for the ULL project. They are:

- KEY_ULL_AIR_PAIRING, to trigger the pairing between the dongle and earbuds or headset.
- KEY_ULL_SWITCH_LINK_MODE, on the headset or earbuds side, to trigger the switch for the link mode between single mode and multi-link mode. The single mode means only connection to either a smartphone or dongle at the same time. The multi-link mode means the DUT can connect to both the smartphone and dongle.
- KEY_ULL_RECONNECT, on the headset or earbuds side, to trigger the switch for the connection between the smartphone and dongle. It is only useful under single mode.

The key mapping table is defined in <project>\src\boards\<Your board>\customerized_key_config.c; Customer can change the table to define the preferred table.

Customer can refer to app_ull_idle_activity.c to review how to process the key events.

3.5.2. Audio Key Actions

Currently, the code uses a rotary key to change the mix ratio and side tone gain. Customer can review the code and implement the feature by the key event.

- Mix ratio:
  Our dongle can implement two audio channels on PC. One channel is for gaming and the other channel is for chatting. Use the key action KEY_AUDIO_MIX_RATIO_GAME_ADD and KEY_AUDIO_MIX_RATIO_CHAT_ADD to process the requirement. The default setting is 21 levels. When level is gaming max level (default is 0), the gaming ratio is 100%, and the chat ratio is 0. When the level is balanced (default is 10), the gaming ratio and chat ratio is 100%. When the level is chat max level (default is 20), the gaming ratio is 0 and the chat ratio is 100%. Customer can change the

macro ULL_MIX_RATIO_GAME_MAX_LEVEL, ULL_MIX_RATIO_CHAT_MAX_LEVEL and ULL_MIX_RATIO_BALANCED_LEVEL.

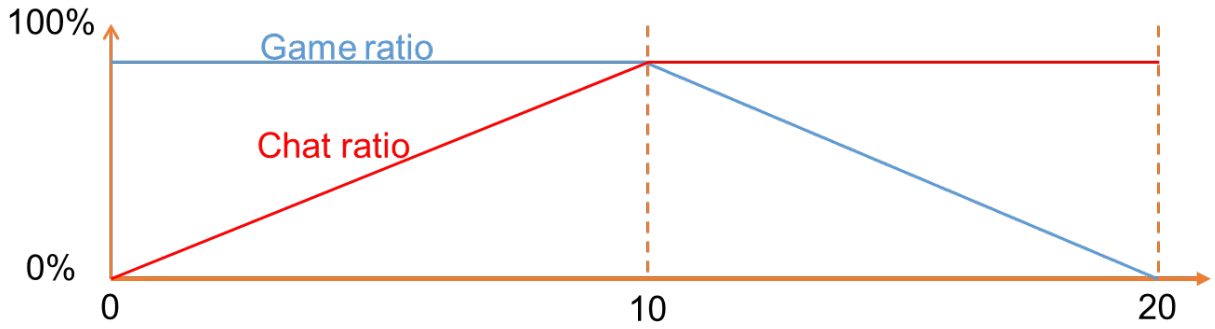

Figure 12. Mix ratio

- Side tone volume:
  Use the KEY_AUDIO_SIDE_TONE_VOLUME_UP and KEY_AUDIO_SIDE_TONE_VOLUME_DOWN to increase or decrease the side tone volume. The minimum value is defined as ULL_SIDE_TOME_VOLUME_MIN_LEVEL; The maximum value is defined as ULL_SIDE_TOME_VOLUME_MAX_LEVEL; The increasing or decreasing value when the user slides the rotary one step is defined as ULL_SIDE_TOME_CHANGE_LEVEL_PRE_STEP.

There is support for pressing a key to mute the microphone. The key action is KEY_MUTE_MIC.

3.5.3. Media Key Actions

Headset or earbuds can control the PC media. The PC media can be ULL connection audio or wired USB audio. The supported actions are KEY_AVRCP_PLAY, KEY_AVRCP_PAUSE, KEY_AVRCP_FORWARD and KEY_AVRCP_BACKWARD. If headset or earbuds have connected with 1 smartphone and 1 PC, and both the smartphone and PC are not playing, the action occurs on smartphone. If PC is playing and smartphone is not playing or is disconnected, the action occurs on PC. The processing code for controlling smartphone media or ULL media is in app_music. The processing code for controlling USB audio media is in app_usb_audio which is only supported by headset.

3.6. PC tool control call of smartphone

When there is a call on smartphone, headset can receive the event and synchronize to PC by race CMD. Customer can check the function app_race_notify_mmi_state() to understand the CMD.
And PC tool have a UI to accept, reject or end call. Customer can check the function bt_race_key_app_event_callback() to understand the CMD.

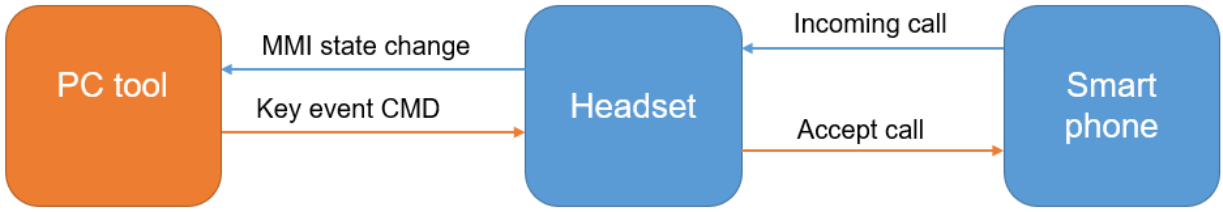

Figure 11. Flow of PC accept the call of smartphone

3.7. AWS Data

Some events related to the ULL must be sent from Agent to Partner or from Partner to Agent.

- Dongle connected flag.
  Event group is EVENT_GROUP_UI_SHELL_APP_INTERACTION, event id is APPS_EVENTS_INTERACTION_ULL_DONGLE_CONNECTION_CHANGE. Device must disable the BLE connection and BLE adv to give enough BT resources for the ULL profile when the dongle is connected so Agent uses the event to notify Partner to disable BLE adv.

- Dongle BT address.
  Event group is EVENT_GROUP_BT_ULTRA_LOW_LATENCY, event id is BT_ULL_EVENT_PAIRING_COMPLETE_IND. Agent sends the BT address of the dongle to Partner.

- Key event.
  Event group is EVENT_GROUP_UI_SHELL_KEY, event id is KEY_ULL_RECONNECT. The Partner cannot handle the reconnect key event so it sends the event to Agent for processing.

3.8. FOTA

When doing FOTA with a smartphone, APP calls bt_cm_write_scan_mode() to disable the page scan.

3.9. Automatically reduce the volume of game audio when chat audio is present

This feature is used for automatically reducing the volume of game audio when chat audio is present. For example, when the dongle detects that there is a human voice on the chat audio channel, it automatically reduces the volume of the game audio channel based on the volume of the chat audio channel as shown in the figure 13. The end-user can then clearly hear the voice of the partner and does not need to worry that the key voice of the partner is drowned out by the game audio.

The settings of this feature are configurable by Airoha Tool as shown in the figure 14.

- Enable/Disable option.
  The user can enable or disable this feature by Enable/Disable option.

- Effective threshold setting.
  This feature is only effective when the volume of the chat audio is equal to or greater than the Effective threshold and the persistent period is equal to or greater than the Effective delay time.

- Effective delay time setting.
  This feature is only effective when the volume of the chat audio is equal to or greater than the Effective threshold and the persistent period is equal to or greater than the Effective delay time.

- Failure threshold setting.
  This feature is ineffective when the volume of the chat audio is equal to or greater than the Failure threshold and the persistent period is equal or greater than the Failure delay time.

- Failure delay time setting.
  This feature is ineffective when the volume of the chat audio is equal to or greater than the Failure threshold and the persistent period is equal to or greater than the Failure delay time.

- Adjustment amount setting.
  This option is used to configure the amount of adjustment relative to the volume of chat audio when the feature is effective. For example, if the volume of chat audio is -5dB and the Adjustment amount setting is -10dB, the volume of game audio will be (-5) + (-10) = -15dB when the feature is effective.

- Ramp up setting.
  This option is used to configure the ramp up step when the feature is ineffective. For example, if the

Ramp up setting is 0.25dB and the volume of chat audio is -5dB and the Adjustment amount setting is -10dB and the volume of game audio is 0dB, the volume of game audio will be increased by 0.25dB gradually at every 1ms till the volume of game audio achieves 0dB from -15dB when the feature is ineffective.

- Ramp down setting.
  This option is used to configure the ramp down step when the feature is effective. For example, if the Ramp down setting is -0.25dB and the volume of chat audio is -5dB and the Adjustment amount setting is -10dB and the volume of game audio is 0dB, the volume of game audio will be decreased by 0.25dB gradually at every 1ms till the volume of game audio achieves -15dB from 0dB when the feature is effective.

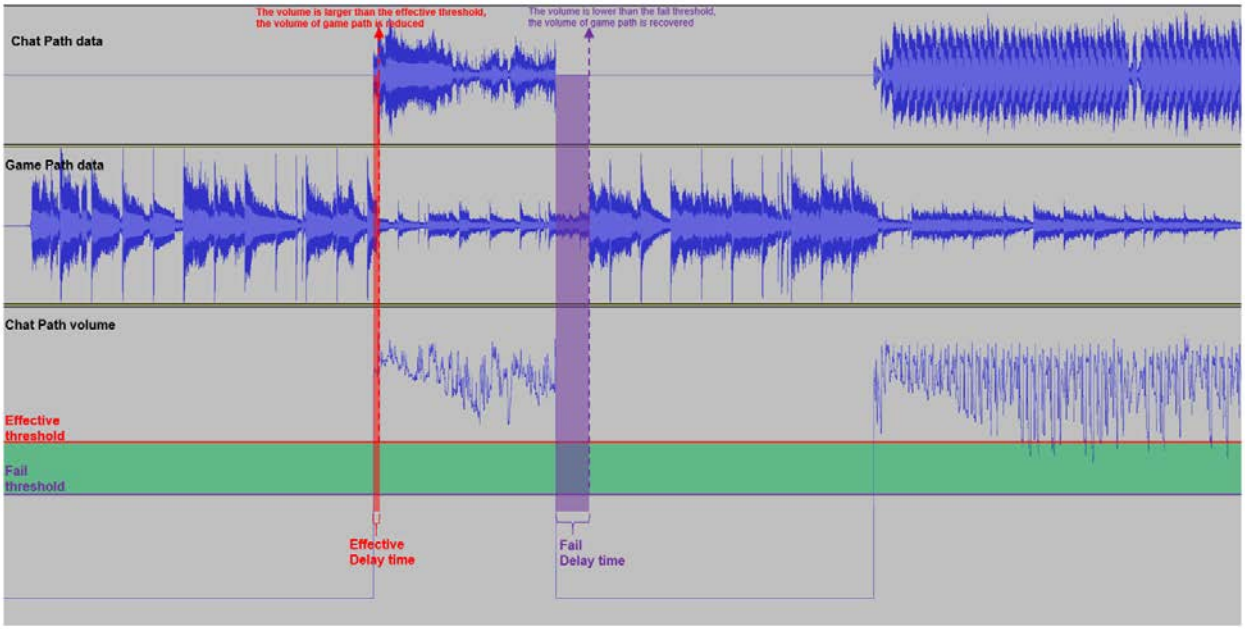

Figure 13. Effect picture

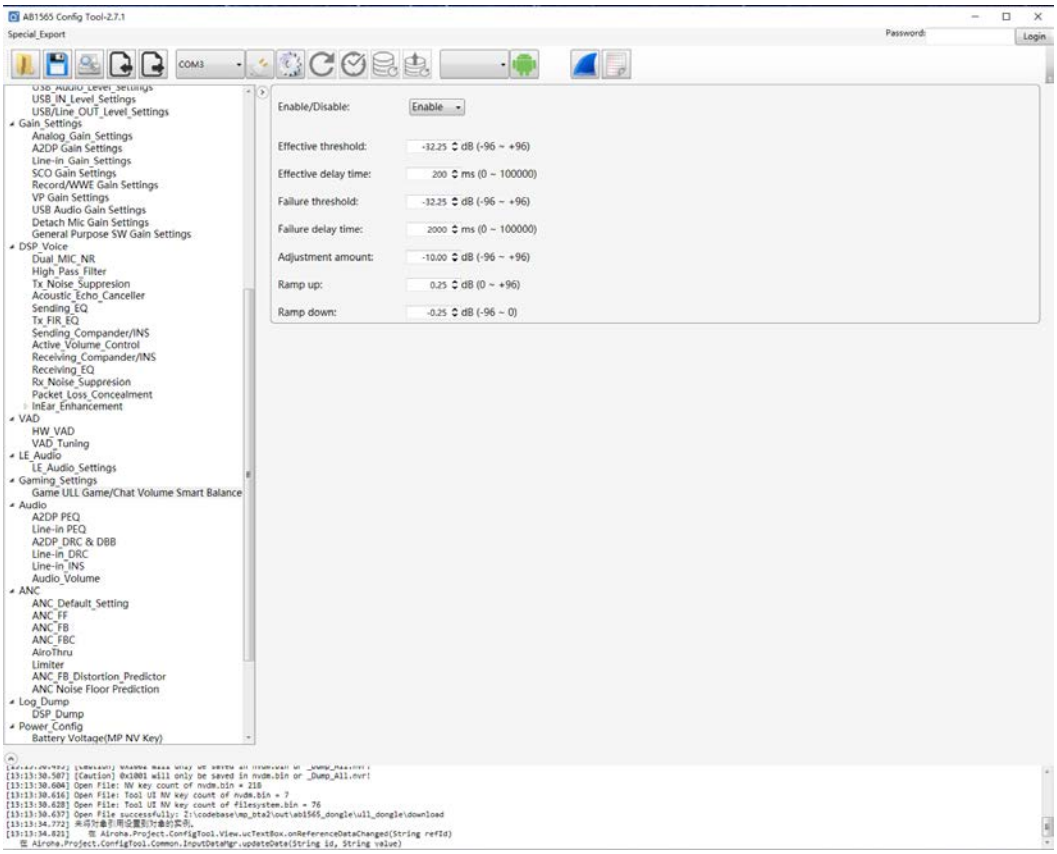

Figure 14. Feature configuration UI