RA4W1 Group

Apple Media Service and Apple Notification Center Service Sample Application

Introduction

This document describes how to create a client role application of Apple Media Service (AMS) and provides a sample application code. The sample application includes Apple Media Service (AMS) and Apple Notification Center Service (ANCS) features. For details of ANCS features, please refer to *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071).

Target Device

- EK-RA4W1

Related Documents

- Bluetooth Core Specification (https://www.bluetooth.com)
- RA4W1 Group User's Manual: Hardware (R01UH0883)
- RA Flexible Software Package User's Manual (R11UM0155)
- Renesas e^2 studio 2020-10 and V7.8 or higher Getting Started Guide (R20UT4891)
- Renesas Flash Programmer V3.08 Flash memory programming software User's Manual (R20UT4813)
- RA4W1 Group Bluetooth LE Profile API Document User's Manual (R11UM0154)
- Bluetooth Low Energy Profile Developer's Guide (R01AN5428)
- EK-RA4W1 Quick Start Guide (R20QS0015)
- RA4W1 Group BLE Sample Application (R01AN5402)
- QE for BLE[RA,RE] V1.2.0 Release Note (R20UT4951EJ)
- RA4W1 Group Apple Notification Center Service Sample Program (R01AN6071)
- Apple Notification Center Service (ANCS) Specification (https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleNotificationCenterServiceSpecification/Introduction/Introduction.html#//apple_ref/doc/uid/TP40013460-CH2-SW1)
- Bundle IDs for native iOS and iPadOS apps in mobile device management (https://support.apple.com/guide/deployment/bundle-ids-for-native-ios-and-ipados-apps-depece748c41/web)
- Apple Media Service Reference (https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleMediaService_Reference/Specification/Specification.html)

Contents

1. Overview

Apple Media Service (Hereafter referred to as AMS) is one of GATT based profile defined by Apple Inc. Devices connected to iOS devices via Bluetooth Low Energy (Bluetooth LE) can use the AMS profile to,

- Control the playback of media on the iOS device.
- Obtain information about the media being played (song title, artist name, etc.)

For details of AMS, please refer to *Apple Media Service Reference*.

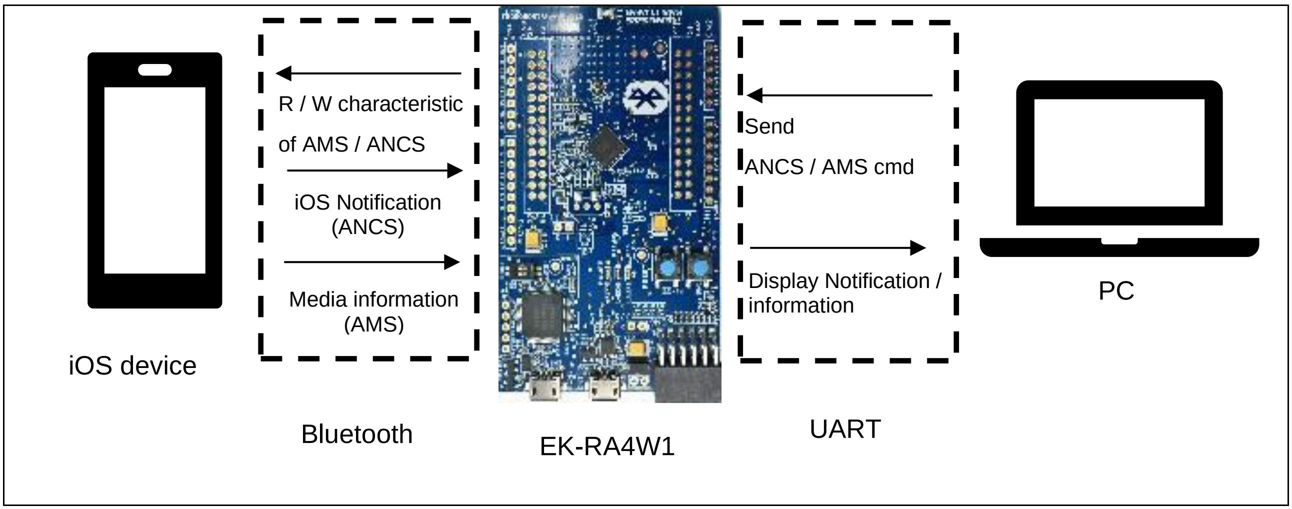


Figure 1. Operating environment

An overview of this sample application is shown in Figure 2. iOS device of AMS server is called Media Source(MS), Bluetooth LE device of AMS client is called Media Remote(MR). To use MS from MR, it is necessary to perform the following procedure. This sample application includes the procedures.

- Pairing.
- Change Client Characteristic Configuration Descriptor (CCCD) to enable notification for each characteristic.

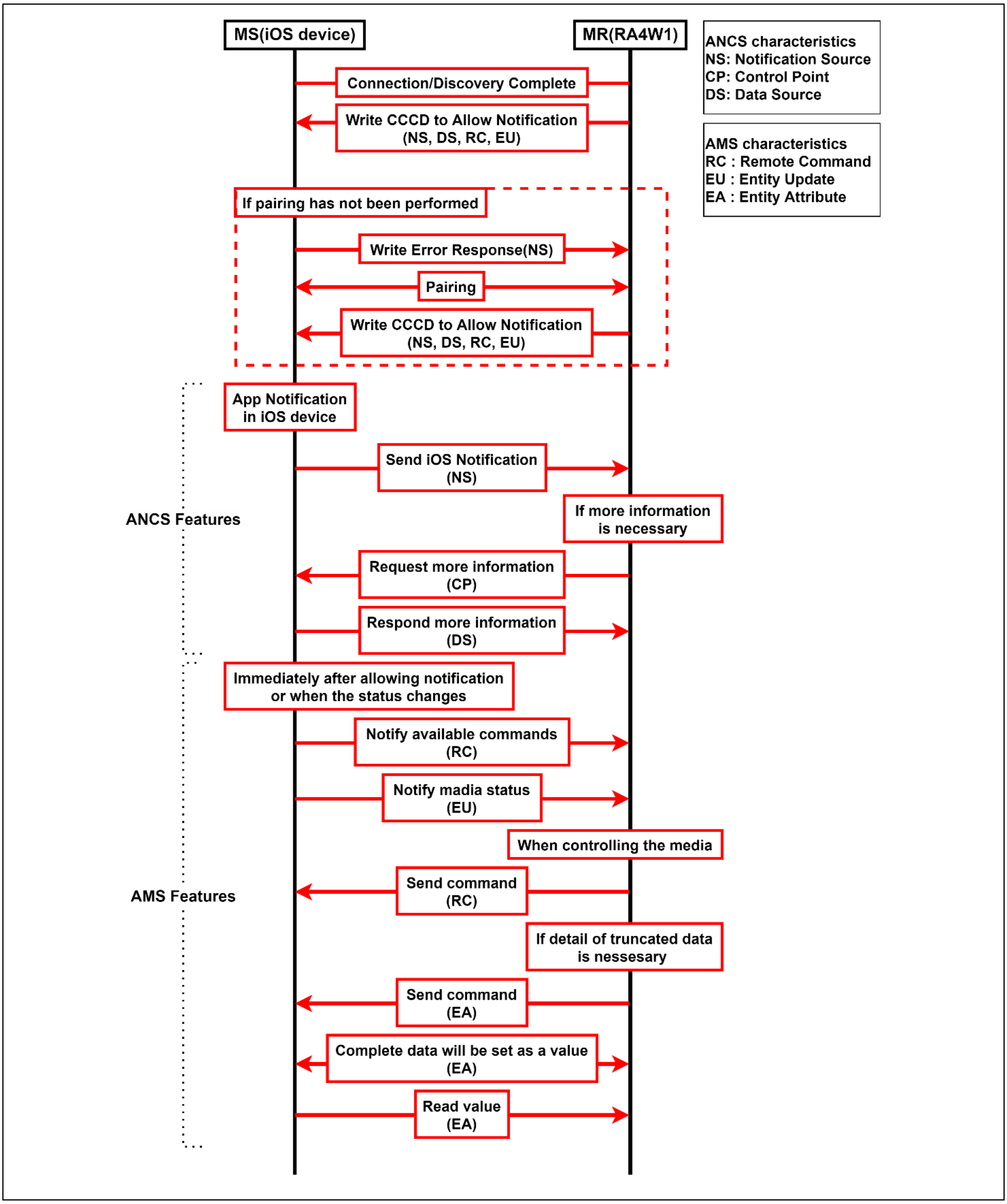

MS(iOS device)
MR(RA4W1)
ANCS characteristics
NS: Notification Source
CP: Control Point
DS: Data Source
AMS characteristics
RC : Remote Command
EU : Entity Update
EA : Entity Attribute
Connection/Discovery Complete
Write CCCD to Allow Notification (NS, DS, RC, EU)
If pairing has not been performed
Write Error Response(NS)
Pairing
Write CCCD to Allow Notification (NS, DS, RC, EU)
ANCS Features
App Notification in iOS device
Send iOS Notification (NS)
If more information is necessary
Request more information (CP)
Respond more information (DS)
AMS Features
Immediately after allowing notification or when the status changes
Notify available commands (RC)
Notify madia status (EU)
When controlling the media
Send command (RC)
If detail of truncated data is nessesary
Send command (EA)
Complete data will be set as a value (EA)
Read value (EA)

Figure 2. Operation flow for this application

1.1 Operation Requirements

Table 1 shows the hardware requirements for building and debugging this sample application.

Table 1. Hardware environment

Hardware	Description
Host PC	Windows® 10 PC with USB interface.
MCU Board	The MCU used must support BLE functions. EK-RA4W1 [RTK7EKA4W1S00000BJ]
On-chip debugging emulators	The EK-RA4W1 has an on-board debugger (J-Link OB). Therefore it is not necessary to prepare an emulator.
E2 lite emulator	Needed if user wants to write device-specific data in the custom board by using Renesas Flash Programmer.
USB cables	Used to connect to the MCU board. EK-RA4W1: 2 USB A-microB cable

Table 2 shows the software requirements for building and debugging BLE software.

Table 2. Software requirements

Software		Version	Description
GCC environment	e² studio	2022-10	Integrated development environment (IDE) for Renesas devices.
	GCC ARM Embedded	10.3.1-2021.10	C/C++ Compiler. (download from e^2 studio installer)
	Renesas Flexible Software Package (FSP)	V4.1.0	Software package for making applications for the RA microcontroller series.
	QE for BLE[RA]	V1.5.0	Generates the source codes (BLE base skeleton program) as a base for the BLE Application and the BLE Profile.
	QE utility [RA]	V1.5.0	
	SEGGER J-Flash	V7.80c	Tool for programming the on-chip flash memory of microcontrollers.
Header files			All API calls and their supporting interface definitions are located in r_ble_api.h and rm_ble_abs_api.h.
Integer types			It uses ANSI C99 “Exact width integer types”. These types are defined in stdint.h.
Endian			Little endian
TeraTerm (VT100 compatible console)			UART Setting Baud rate：115200bps Data：8bit Parity：None Stop bits：None Flow Control：None Character encoding：UTF-8

1.2 Import sample project

This project is provided in the form of a zip file and can be imported into e2studio. For import procedures, please refer to the [*Renesas e2 studio 2020-10 and V7.8 or higher Getting Started Guide (R20UT4891)*]. After importing sample projects, open FSP configurator, press “Generate Project Contents” button and re-generate FSP module which uses the sample project. The zip file can be imported is as follows.

- ble_baremetal_ek_ra4w1_ams_client
- ble_freertos_ek_ra4w1_ams_client

2. Operation of the sample application

This chapter describes the operating procedure of this sample application.

2.1 Application Option

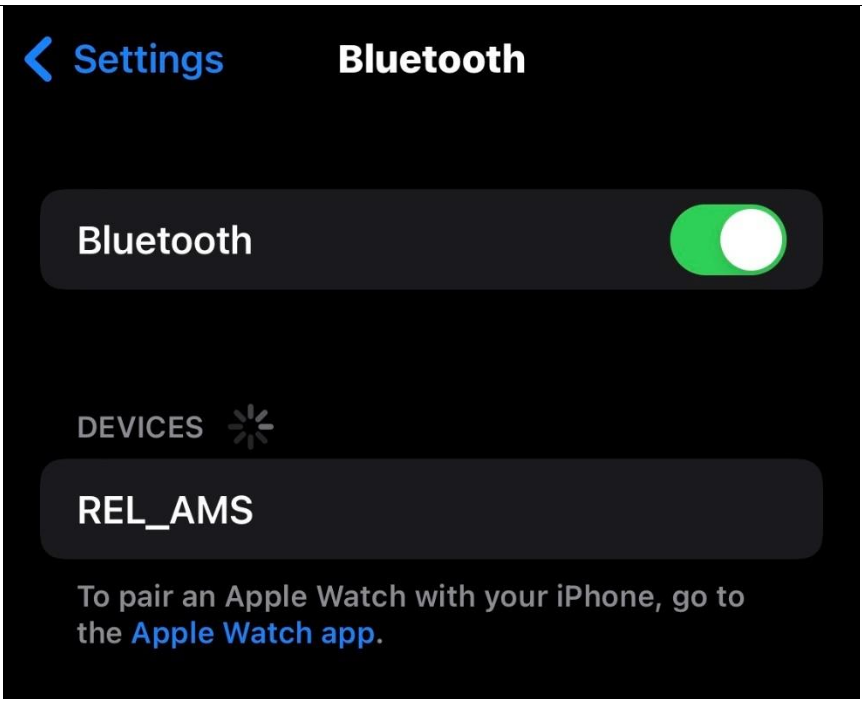
Please refer to *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071).

2.2 Operation Flow of sample application

This section describes the procedure for connection with iOS device and AMS data communication after running the sample application.

2.2.1 Connection with iOS Device

Please refer to *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071). When you run this sample application, it will be displayed with the name *"REL_AMS"* on iOS device Bluetooth Setting screen.



Bluetooth

REL_AMS

To pair an Apple Watch with your iPhone, go to the Apple Watch app.

Figure 3. Device name

When the connection with iOS device has established and data communication is possible, the following text will be displayed in the console.

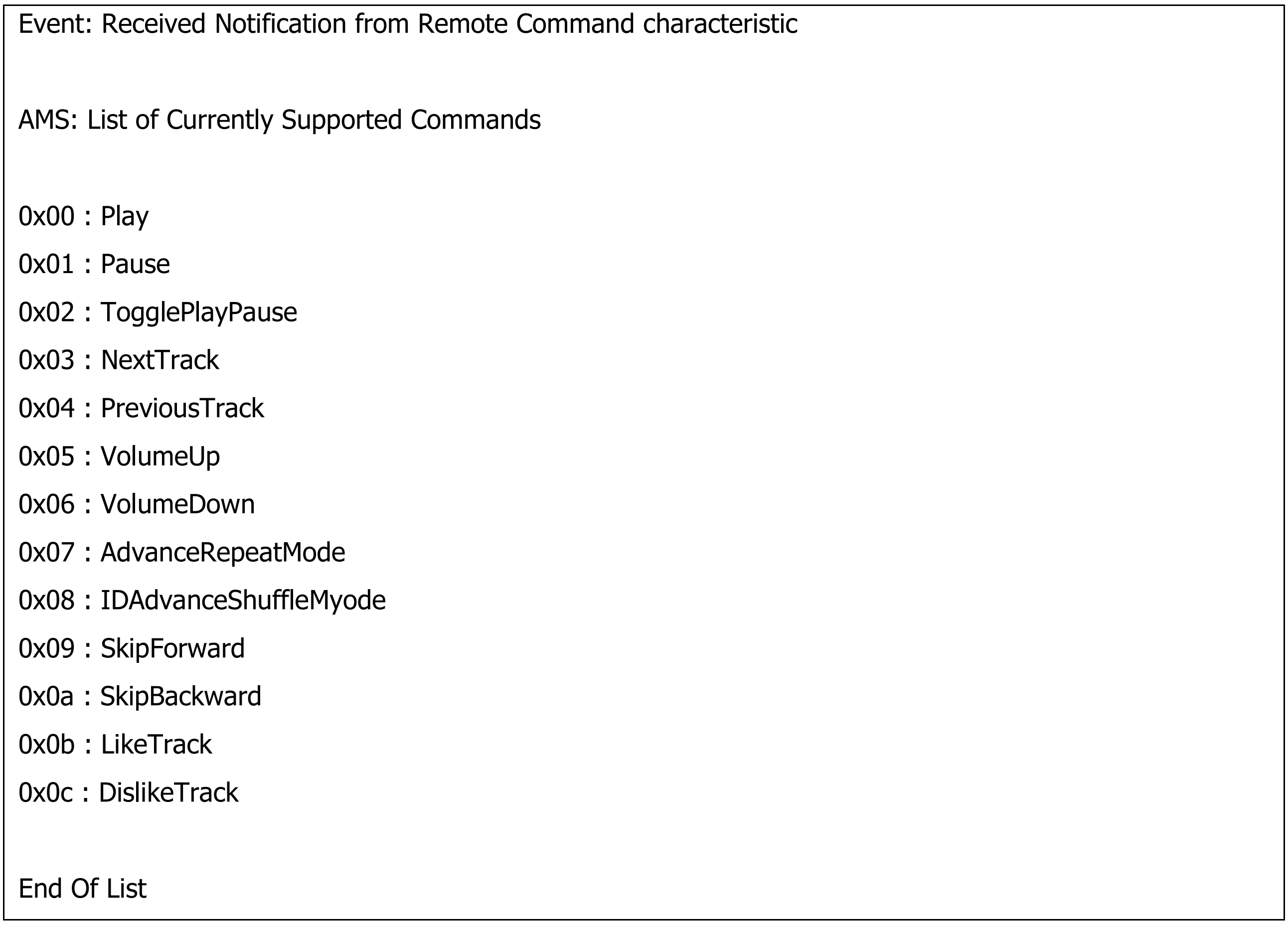
```
ANCS : Notification Enabled
receive BLE_AMSC_EVENT_RC_CLI_CNFG_WRITE_RSP
receive BLE_AMSC_EVENT_RC_HDL_VAL_NTF
```

Figure 4. Ready for ANCS and AMS communication

2.2.2 Remote Command characteristic features

2.2.2.1 Receive supported remote command list from MS (iOS device)

MS (iOS device) will send a supported remote command list from Remote Command characteristic as GATT notification when the user launches or closes iOS application (e.g. Music App). Received the command list will be shown in the console. Figure 5 shows an example of a command list from MS (iOS device) when the user launches Music App.



```
Event: Received Notification from Remote Command characteristic


AMS: List of Currently Supported Commands


0x00 : Play
0x01 : Pause
0x02 : TogglePlayPause
0x03 : NextTrack
0x04 : PreviousTrack
0x05 : VolumeUp
0x06 : VolumeDown
0x07 : AdvanceRepeatMode
0x08 : IDAdvanceShuffleMyode
0x09 : SkipForward
0x0a : SkipBackward
0x0b : LikeTrack
0x0c : DislikeTrack


End Of List
```

Figure 5. Supported remote command list from MS(iOS device) when lunch the Music App

2.2.2.2 Display supported remote command stored in MR (RA4W1)

MR (RA4W1) will store a supported command list which received in section 2.2.2.1. User can display the stored command list by using *"amsc disp_rc"* command as follows. Please refer to section 3.3 about usage of *"amsc disp_rc"* command.

```
$amsc disp_rc
AMS: List of Currently Supported Commands


0x00 : Play
0x01 : Pause
0x02 : TogglePlayPause
0x03 : NextTrack
0x04 : PreviousTrack
0x05 : VolumeUp
0x06 : VolumeDown
0x07 : AdvanceRepeatMode
0x08 : IDAdvanceShuffleMyode
0x09 : SkipForward
0x0a : SkipBackward
0x0b : LikeTrack
0x0c : DislikeTrack


End Of List
```

Figure 6. Display the supported remote command list stored in MR (RA4W1)

2.2.2.3 Control media playback of MS (iOS device) by sending a remote command

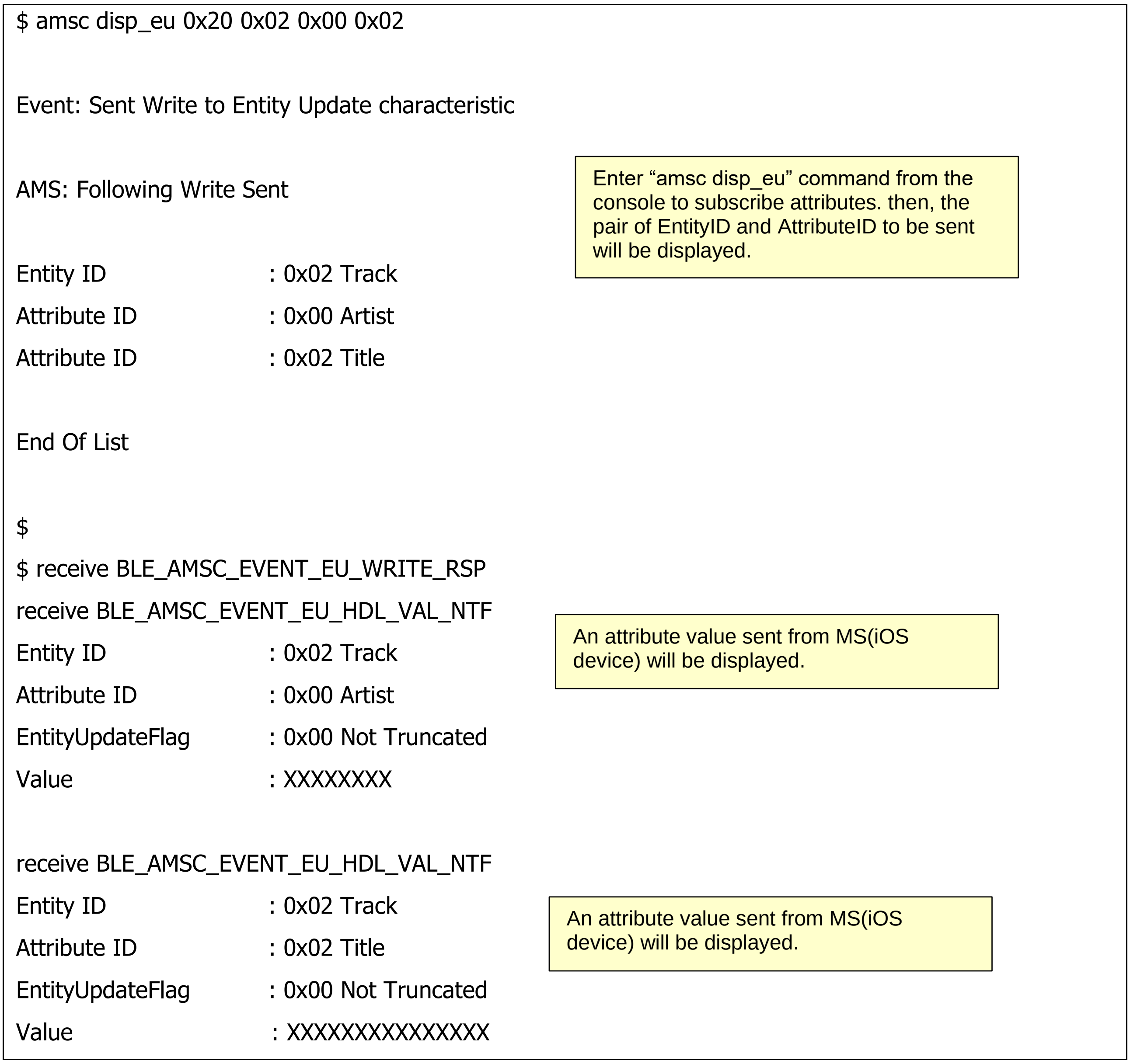
The user can control media playback of MS(iOS device) such as play, pause, volume up / down by sending a remote command to Remote Command characteristic of MS (iOS device). The remote command consists of a RemoteCommandID which specified a media command to be executed. For details of RemoteCommandID, please refer to *Apple Media Service Reference*. To send the remote command, use *"amsc send_rc"* command from the console. When the user enters *"amsc send_rc"* command, the RemoteCommandID to be sent will be displayed. Regardless of the remote command is successfully executed or not, MS (iOS device) will not send a result. Please refer to section 3.3 about the usage of *"amsc send_rc"* command,

```
$ amsc send_rc 0x20 0x01

Event: Sent Write to Remote Command characteristic

AMS: Following Command Sent

0x01: Pause

End Of List
```

Figure 7. send remote command

2.2.3 Entity Update characteristic

The user can receive media information such as a track title and artist name from MS (iOS device) by subscribing to attribute(s). An attribute consists of EntityID and AttributeID. To subscribe to attribute(s), use *“amsc disp_eu”* command and specify EntityID / AttributeID pair as argument of the command. Attributes that have the same EntityID can subscribe simultaneously. The example shown in Figure 8 subscribes an artist name (EntityID : 0x02 Track, AttributeID : Artist) and a track title (EntityID: 0x02 Track, AttributeID: Title). After subscribing attribute(s), MS (iOS device) will send attribute(s) value as GATT notification when attribute(s) has changed. Please refer to section 3.3 about usage of *“amsc disp_eu”* command.



```
$ amsc disp_eu 0x20 0x02 0x00 0x02

Event: Sent Write to Entity Update characteristic

AMS: Following Write Sent

Entity ID                   : 0x02 Track
Attribute ID                : 0x00 Artist
Attribute ID                : 0x02 Title

End Of List

$
$ receive BLE_AMSC_EVENT_EU_WRITE_RSP
receive BLE_AMSC_EVENT_EU_HDL_VAL_NTF
Entity ID                   : 0x02 Track
Attribute ID                : 0x00 Artist
EntityUpdateFlag            : 0x00 Not Truncated
Value                       : XXXXXXXX

receive BLE_AMSC_EVENT_EU_HDL_VAL_NTF
Entity ID                   : 0x02 Track
Attribute ID                : 0x02 Title
EntityUpdateFlag            : 0x00 Not Truncated
Value                        : XXXXXXXXXXXXXXX
```

Enter “amsc disp_eu” command from the console to subscribe attributes. then, the pair of EntityID and AttributeID to be sent will be displayed.

An attribute value sent from MS(iOS device) will be displayed.

An attribute value sent from MS(iOS device) will be displayed.

Figure 8. Subscribing attributes

2.3 Directory/File Structure

Table 3 shows the directory / file structure of this application.

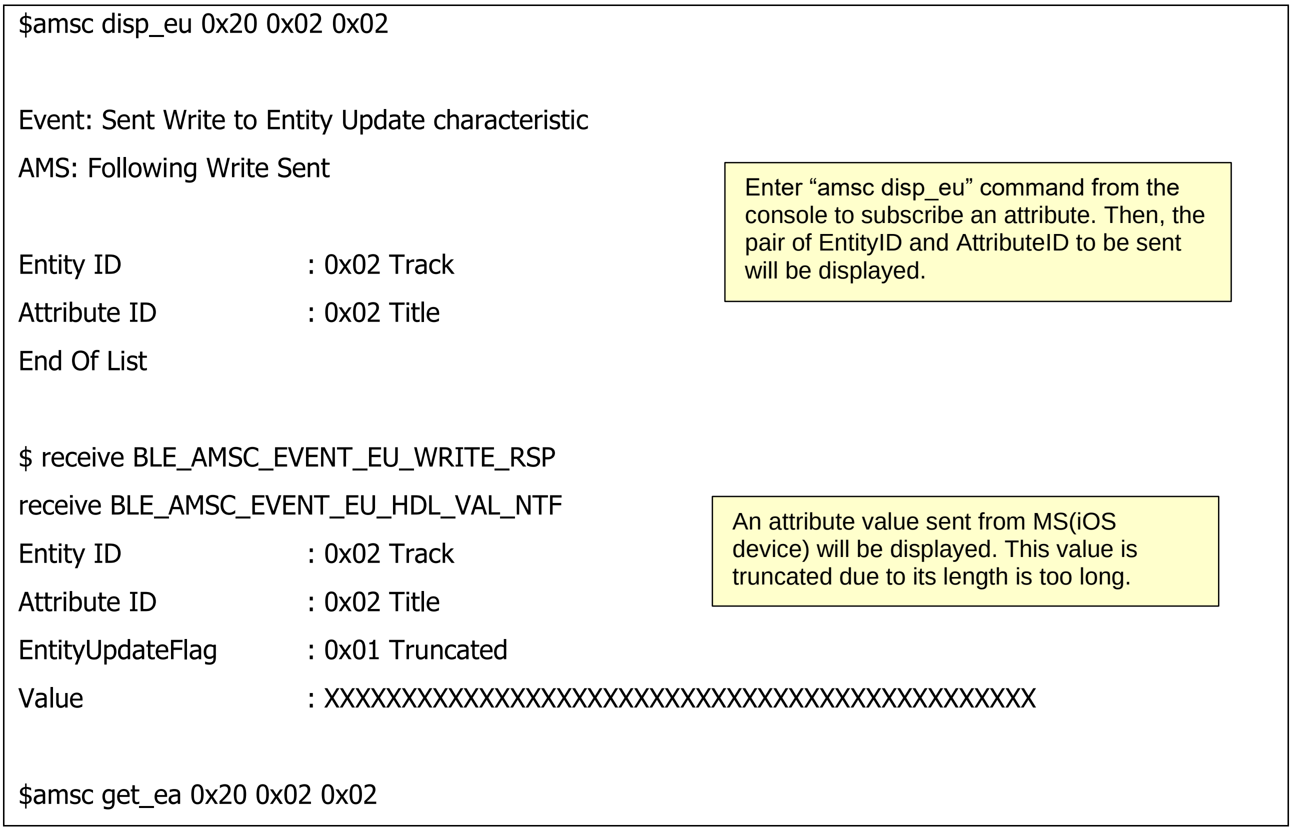
Table 3. Directory / File structure

Directory/File structure			Description
qe_gen\	ble\	app_main.c	BLE application main code
		gatt_db.c gatt_db.h	GATT database code
		r_ble_ANCSc.c r_ble_ANCSc.h	ANCS profile code
		r_ble_AMSc.c r_ble_AMSc.h	AMS profile code
		discovery\	Discovery operation library
		profile_cmn\	Profile operation library
src\	hal_entry.c		project main code
	app_lib		Library for using command line function.
	ble_core_task_entry.c [*1]		Bluetooth LE task code for FreeRTOS project
	ams_client.json		QE for BLE json file for AMS client role
	ancs_client.json		QE for BLE json file for ANCS client role

*1 This file is included only in FreeRTOS project

2.3.1 Entity Attribute characteristic

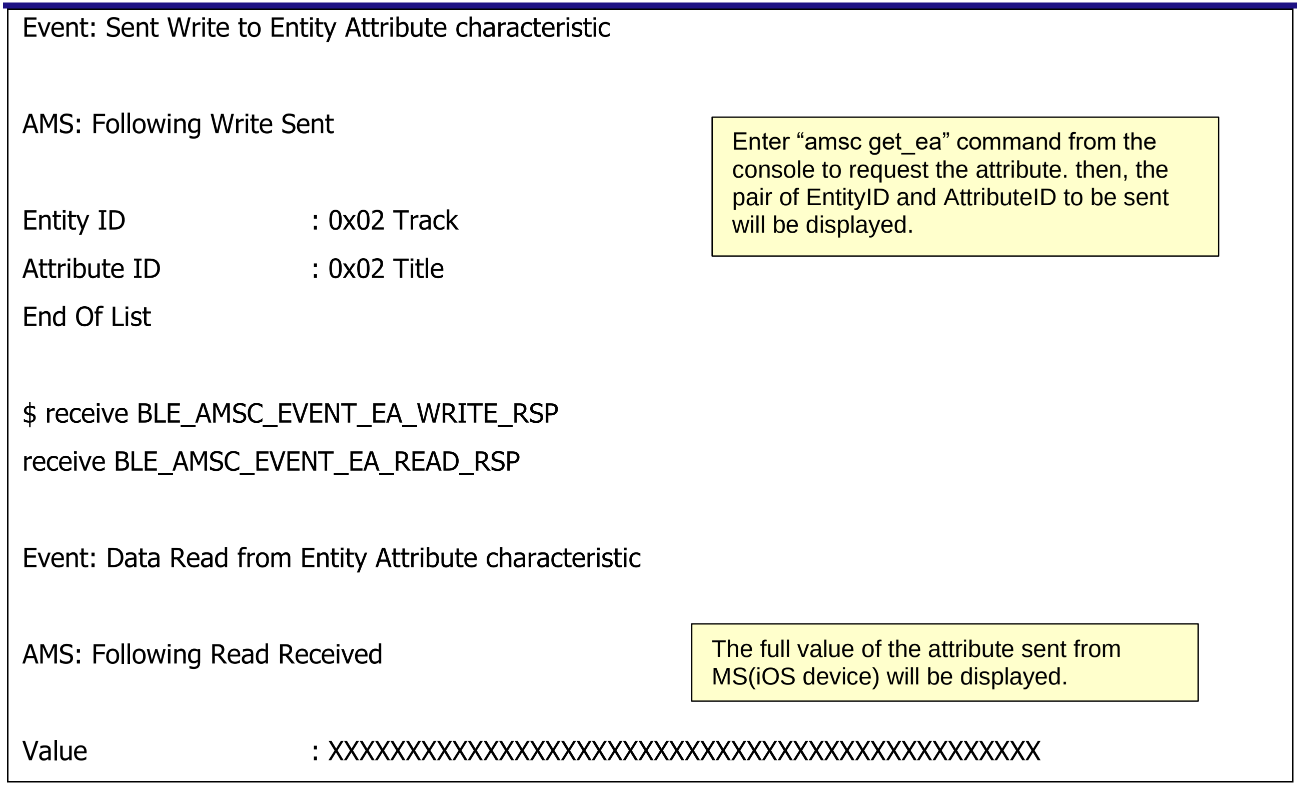
The maximum length of attribute value from MS (iOS device) depends on MTU size. When length of attribute value is longer than MTU size, the attribute value will be truncated. The user can request the full value of the attribute value by using *"amsc get_ea"* command. The example shown in Figure 9 when the user subscribes a track title (EntityID: 0x02 Track, AttributeID: 0x02 Title) by using *"amsc disp_eu"* command, receives truncated value and requests the full value of the attribute by using *"amsc get_ea"* command. Please refer to section 3.3 about the usage of *"amsc get_ea"* command.



```
$amsc disp_eu 0x20 0x02 0x02


Event: Sent Write to Entity Update characteristic
AMS: Following Write Sent


Entity ID            : 0x02 Track
Attribute ID         : 0x02 Title
End Of List


$ receive BLE_AMSC_EVENT_EU_WRITE_RSP
receive BLE_AMSC_EVENT_EU_HDL_VAL_NTF
Entity ID            : 0x02 Track
Attribute ID         : 0x02 Title
EntityUpdateFlag     : 0x01 Truncated
Value                : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


$amsc get_ea 0x20 0x02 0x02
```

Enter "amsc disp_eu" command from the console to subscribe an attribute. Then, the pair of EntityID and AttributeID to be sent will be displayed.

An attribute value sent from MS(iOS device) will be displayed. This value is truncated due to its length is too long.



```
Event: Sent Write to Entity Attribute characteristic

AMS: Following Write Sent

Entity ID                : 0x02 Track
Attribute ID             : 0x02 Title
End Of List

$ receive BLE_AMSC_EVENT_EA_WRITE_RSP
receive BLE_AMSC_EVENT_EA_READ_RSP

Event: Data Read from Entity Attribute characteristic

AMS: Following Read Received

Value                    : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Enter “amsc get_ea” command from the console to request the attribute. then, the pair of EntityID and AttributeID to be sent will be displayed.

The full value of the attribute sent from MS(iOS device) will be displayed.

Figure 9. Request full value of a truncated attribute

3. Program description

This chapter describes the program and implementation of this application.

3.1 Services

Table 4 shows UUIDs and attribute properties of AMS and ANCS included with this sample application.

Table 4. Service information of AMS and ANCS

Service	**UUID**	**Properties**
Apple Media Service	89D3502B-0F36-433A-8EF4-C502AD55F8DC	---
Apple Notification Center Service	7905F431-B5CE-4E99-A40F-4B1E122D00D0	---
Characteristic	**UUID**	**Properties**
Remote Command (AMS)	9B3C81D8-57B1-4A8A-B8DF-0E56F7CA51C2	writeable, notifiable
Entity Update (AMS)	2F7CABCE-808D-411F-9A0C-BB92BA96C102	writeable with response, notifiable
Entity Attribute (AMS)	C6B2F38C-23AB-46D8-A6AB-A3A870BBD5D7	readable, writeable
Notification Source (ANCS)	9FBF120D-6301-42D9-8C58-25E699A21DBD	notifiable
Control Point (ANCS)	69D1D8F3-45E1-49A8-9821-9BBDFDAAD9D9	writeable
Data Source (ANCS)	22EAC6E9-24D6-4BB5-BE44-B36ACE7C7BFB	notifiable

For more details of each service and characteristic of AMS and ANCS, please refer to *Apple Media Service Reference* and *Apple Notification Center Service (ANCS) Specification.*

3.2 API reference

3.2.1 AMS service APIs

This section describes the APIs, a list of errors, the data structure of AMS services.

Initialize function		
Definition:	ble_status_t R_BLE_AMSC_Init(ble_servc_app_cb_t cb)	
Description:	Initialize AMS and register a callback function. AMS characteristic related events will be notified to the registered callback function.	
Argument:	cb	Callback function to be registered

Service discovery call function		
Definition:	R_BLE_AMSC_ServDiscCb(uint16_t conn_hdl, uint8_t serv_idx, uint16_t type, void *p_param)	
Description:	AMS callback function for service discovery procedure. This function is used as one of argument of *R_BLE_DISC_Start* API.	
Argument:	conn_hdl	Connection handle
	serv_idx	Service index
	type	Event type
	p_param	Pointer to event parameter

Get attribute handle function		
Definition:	R_BLE_AMSC_GetServAttrHdl(const st_ble_dev_addr_t *p_addr, st_ble_gatt_hdl_range_t *p_hdl)	
Description:	Gets attribute handle of AMS.	
Argument:	p_addr	Pointer to BD address of target device to get attribute handle
	p_hdl	Pointer to store attribute handle

When an error occurs in GATT communication, *BLE_GATTC_EVENT_ERROR_RSP* event is notified to the GATT client callback function with the error code as event data. The error codes related to the AMS service are shown in Table 5.

Table 5. AMS error code

Error code	Description
BLE_AMSC_INVALID_STATE_ERROR Value: 0xA0	MR is not set up AMS properly. For example, writing to Entity Update or Entity Attribute performed without subscribing to the GATT notification of Entity Update characteristic.
BLE_AMSC_INVALID_COMMAND_ERROR Value: 0xA1	The format of the written command is invalid.
BLE_AMSC_ABSENT_ATTRIBUTE_ERROR Value: 0xA2	The corresponding Attribute is empty. For example, this error happens when you play a track that has no artist information and request artist information in the Entity Attribute.

3.2.2 Remote Command Characteristic APIs and event

This section describes APIs, a list of events, a data structure of the Remote Command characteristic.

Get attribute handle function		
Definition :	R_BLE_AMSC_GetRcAttrHdl(const st_ble_dev_addr_t *p_addr, st_ble_AMSc_rc_attr_hdl_t *p_hdl)	
Description :	Gets attribute handle of Remote Command characteristic on MS (iOS device).	
Argument :	p_addr	Pointer to BD address of target device to get attribute handle
	p_hdl	Pointer to store attribute handle

Read function for Client Characteristic Configuration Descriptor		
Definition :	R_BLE_AMSC_ReadRcCliCnfg(uint16_t conn_hdl)	
Description :	Send Read Request to CCCD of Remote Command characteristic on MS (iOS device). *BLE_AMSC_EVENT_RC_CLI_CNFG_READ_RS* event will be notified when Read Response is received from MS (iOS device).	
Argument :	conn_hdl	Connection handle

Write function for Client Characteristic Configuration Descriptor		
Definition :	R_BLE_AMSC_WriteRcCliCnfg(uint16_t conn_hdl, const uint16_t *p_value)	
Description :	Send Write Request to CCCD of Remote Command characteristic on MS (iOS device). To receive Notification from Remote Command characteristic, it is necessary to write “1” to the CCCD by using this API. *BLE_AMSC_EVENT_RC_CLI_CNFG_WRITE_RSP* event will be notified when write response received from MS (iOS device) when write response received.	
Argument :	conn_hdl	Connection handle
	p_value	Pointer to data sending with Write Request

The following events occur when using the above APIs.

Table 6. Remote Command events

Event	**Description**
BLE_AMSC_EVENT_RC_WRITE_RSP	Receive Write Response of Remote Command characteristic. Data: None
BLE_AMSC_EVENT_RC_HDL_VAL_NTF	Receive notification from Remote Command characteristic. Data: *st_ble_seq_data_t*
BLE_AMSC_EVENT_RC_CLI_CNFG_READ_RSP	Receive Read Response of CCCD included in Remote Command characteristic. Data: uint16_t

The length of the supported command list from an MS (iOS device) is variable. Therefore, the data structure of Remote Command has variable length as *st_ble_seq_data_t* type. It is necessary to decode in the application.

3.2.3 Entity Update Characteristic

This section describes APIs, a list of events, a data structure of the Entity Update characteristic.

Get attribute handle function		
Definition:	void R_BLE_AMSC_GetEuAttrHdl(const st_ble_dev_addr_t *p_addr, st_ble_AMSc_eu_attr_hdl_t *p_hdl)	
Description	Get the attribute handle of Entity Update characteristic on MS (iOS device).	
Argument:	p_addr	Pointer to BD address of target device to get attribute handle
	p_hdl	Pointer to store attribute handle

Read function of Client Characteristic Configuration Descriptor		
Definition:	ble_status_t R_BLE_AMSC_ReadEuCliCnfg(uint16_t conn_hdl)	
Description	Send Read Request to the CCCD of Entity Update characteristic on MS (iOS device). *BLE_AMSC_EVENT_EU_CLI_CNFG_READ_RSP* event will be notified when Read Response is received from MS (iOS device).	
Argument:	conn_hdl	Connection handle

Write function of Client Characteristic Configuration Descriptor		
Definition:	ble_status_t R_BLE_AMSC_WriteEuCliCnfg(uint16_t conn_hdl, const uint16_t *p_value)	
Description	Send Write Request to CCCD of Entity Update characteristic on MS (iOS device). To receive Notification from Entity Update characteristic, it is necessary to write “1” to the CCCD by using this API. *BLE_AMSC_EVENT_EU_CLI_CNFG_WRITE_RSP* event will be notified when write response received.	
Argument:	conn_hdl	Connection handle
	p_value	Pointer to data sending with Write Request

Write function		
Definition:	ble_status_t R_BLE_AMSC_WriteEu(uint16_t conn_hdl, const st_ble_seq_data_t *p_value);	
Description	Send a Write Request to Entity Update characteristic on MS (iOS device). *BLE_AMSC_EVENT_EU_WRITE_RSP* event will be notified when write response received.	
Argument:	conn_hdl	Connection handle
	p_value	Pointer to data sending with Write Request

The following events will be notified when entity update characteristic related APIs called.

Table 7. Entity Update Event

Event	**Description**
BLE_AMSC_EVENT_EU_WRITE_RSP	Receive Write Response of Entity Update characteristic. Data: None
BLE_AMSC_EVENT_EU_HDL_VAL_NTF	Receive Notification of Entity Update characteristic. Data: *st_ble_seq_data_t*
BLE_AMSC_EVENT_EU_CLI_CNFG_READ_RSP	Receive Read Response of CCCD included Entity Update characteristic. Data: uint16_t
BLE_AMSC_EVENT_EU_CLI_CNFG_WRITE_RSP	Receive Write Response of CCCD included in Entity Update characteristic. Data: none

The argument of *R_BLE_AMSC_WriteEu* API has variable length and its type is *st_ble_seq_data_t* structure. And *BLE_AMSC_EVENT_EU_HDL_VAL_NTF* event data also has variable length and the type is *st_ble_seq_data_t* structure as well. Therefore, these kinds of variables should be decoded by the application when used it.

3.2.4 Entity Attribute Characteristic

This section describes APIs, a list of events, a data structure of Entity Attribute characteristic.

Get attribute handle function		
Definition:	R_BLE_AMSC_GetEaAttrHdl(const st_ble_dev_addr_t *p_addr, st_ble_AMSc_ea_attr_hdl_t *p_hdl)	
Description:	Gets attribute handle of Entity Attribute characteristic on MS (iOS device).	
Argument:	p_addr	Pointer to BD address of target device to get attribute handle
	p_hdl	Pointer to store attribute handle

Read function		
Definition:	ble_status_t R_BLE_AMSC_ReadEa(uint16_t conn_hdl)	
Description:	Send a Read Request to Entity Attribute characteristic on MS (iOS device). *BLE_AMSC_EVENT_EA_READ_RSP* event will be notified when read response received.	
Argument:	conn_hdl	Connection handle

Write function		
Definition:	ble_status_t R_BLE_AMSC_WriteEa(uint16_t conn_hdl, const st_ble_seq_data_t *p_value);	
Description:	Send a Write Request to Entity Attribute characteristic on MS (iOS device). *BLE_AMSC_EVENT_EA_WRITE_RSP* event will be notified when write response received.	
Argument:	conn_hdl	Connection handle
	p_value	Pointer to data sending with Write Request

The following events will be notified when entity attribute characteristic related APIs called.

Table 8. Entity Attribute Event

Event	**Description**
BLE_AMSC_EVENT_EA_READ_RESP	Receive Read Response of Entity Attribute characteristic. Data: *st_ble_seq_data_t*
BLE_AMSC_EVENT_EA_WRITE_RESP	Receive Write Response for Entity Attribute characteristic. Data: None

BLE_AMSC_EVENT_EA_READ_RESP event data has a variable length and the type is *st_ble_seq_data_t*. Therefore, these kinds of variables should be decoded by the application when used it.

3.3 Command reference

This section describes the command defined in this application. These commands can be used by typing them into the console after connecting with iOS device.

disp_rc command	
Format:	amsc disp_rc
Description:	Print to console the list of supported Remote Commands that MR (RA4W1) last received from MS (iOS device).
Parameter:	None
Example:	$ amsc disp_rc

send_rc command		
Format:	amsc send_rc [conn_hdl] [RemoteCommandID]	
Description:	Write Remote Command to MS (iOS device). The MS will change a media playback state according to received command.	
Parameter:	[conn_hdl]	Specifies the connection handle of the MS (iOS device) to which the Write Request will be sent.
	[RemoteCommandID]	Specify the RemoteCommandID that you want MS (iOS device) to execute. For a list of RemoteCommandID, please refer to *Apple Media Service Reference*.
Example:	$ amsc send_rc 0x0020 0x01 Sends Pause = 0x01 as Remote Command to MS (iOS device). MS (iOS device) will stop playing the media.	

amsc disp_eu command		
Format:	amsc disp_eu [conn_hdl] [EntityID] {AttributeIDs}	
Description:	Send the Entity Update command to MS(iOS device) to subscribe attribute(s).	
Parameter:	[conn_hdl]	Specifies the connection handle of the MS (iOS device) to be sent the command.
	[EntityID]	Specifies the EntityID of attribute(s) to be subscribed.
	[AttributeIDs]	Specify AttributeID(s) to be subscribed. Attributes that have the same Entity ID can subscribe simultaneously.
Example:	$ amsc disp_eu 0x0020 0x02 0x00 0x02 Sends EntityIDTrack = 0x02, TrackAttributeIDArtist = 0x00 as Entity update command to MS (iOS device). MS(iOS device) will notify Artist and Title attributes.	

amsc get_ea command		
Format:	amsc get_ea [conn_hd] [EntityID] {AttributeID}	
Description:	Send the Entity Attribute command to request the full value of the attribute.	
Parameter	[conn_hdl]	Specifies the connection handle of the MS (iOS device) to be sent the command.
	[EntityID]	Specifies the EntityID of the attribute to be requested.
	[AttributeID]	Specify the AttributeID of the attribute to be requested.
Example	$ amsc get_ea 0x0020 0x02 0x02 Sends EntityIDTrack = 0x02, TrackAttributeIDArtist = 0x20 as Entity attribute command to MS (iOS device). MS(iOS device) will notify full value of Artist and title attributes.	

3.4 Implementation

The sample application provides an example to connect with iOS device using RA4W1 and perform data communication of AMS client role over Bluetooth LE communication. This section describes the behavior and implementation of this sample application program. This sample application also includes the ANCS part. For the ANCS part, please refer to *RA4W1 Group Apple Notification Center Service Sample Program (R01AN6071)*.

3.4.1 Initialization

Initialization of AMS is executed by *R_BLE_AMSC_Init* API. This API is called in *ble_init()* function defined in *app_main.c*. The user does not need additional processing.

```
static ble_status_t ble_init(void)
{
    ble_status_t status;
………
    /* Initialize Apple Media Service client API */
    status = R_BLE_AMSC_Init(AMSc_cb);
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }
    return status;
}
```

Code 1. Initialization

3.4.2 Enable Resolvable Private Address function

Implementation is same as *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071). Please refer to it.

3.4.3 Start Advertising

Implementation is same as *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071). Please refer to it.

3.4.4 Service Discovery

Implementation is same as *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071). Please refer to it.

3.4.5 Pairing and enabling notification

(1) Pairing

Please refer to the *RA4W1 Group Apple Notification Center Service Sample Program*(R01AN6071) for the pairing method.

(2) Enabling notification

Enable notification for characteristics which have notification property in the following order.

```
static void ANCSc_cb(uint16_t type, ble_status_t result, st_ble_servc_evt_data_t *p_data)
{
    switch(type)
    {
.........
       case BLE_ANCSC_EVENT_NS_CLI_CNFG_WRITE_RSP:
       {
          if (BLE_SUCCESS == result)
          {
              /* Enable Notification of Data Source */
              uint16_t cccd_value = BLE_GATTS_CLI_CNFG_NOTIFICATION;
              R_BLE_ANCSC_WriteDsCliCnfg(p_data->conn_hdl, &cccd_value);
          }
       } break;

       case BLE_ANCSC_EVENT_DS_CLI_CNFG_WRITE_RSP:
       {
          if (BLE_SUCCESS == result)
          {
               /* ANCS Notification is Enabled */
               R_BLE_CLI_Printf("ANCS : Notification Enabled \n");
               uint16_t cccd_value = BLE_GATTS_CLI_CNFG_NOTIFICATION;
               R_BLE_AMSC_WriteRcCliCnfg(p_data->conn_hdl, &cccd_value);
          }
       } break;
.........
    }
}

static void AMSc_cb(uint16_t type, ble_status_t result, st_ble_servc_evt_data_t *p_data)
{
    switch(type)
    {
       case BLE_AMSC_EVENT_RC_CLI_CNFG_WRITE_RSP:
       {
           R_BLE_CLI_Printf("receive BLE_AMSC_EVENT_RC_CLI_CNFG_WRITE_RSP\n");

           /* Enable Notification of Entity Update */
           uint16_t cccd_value = BLE_GATTS_CLI_CNFG_NOTIFICATION;
           R_BLE_AMSC_WriteEuCliCnfg(p_data->conn_hdl, &cccd_value);
           break;
       }
......
       case BLE_AMSC_EVENT_EU_CLI_CNFG_WRITE_RSP:
       {
           R_BLE_CLI_Printf("receive BLE_AMSC_EVENT_EU_CLI_CNFG_WRITE_RSP\n");
           break;
       }
......
    }
}
```

1. When writing to CCCD of Notification Source(ANCS) is completed, start writing to CCCD of Data Source(ANCS).
2. When writing to CCCD of Data Source(ANCS) is completed, start writing to CCCD of Remote Command(AMS).
3. When writing to CCCD of Remote Command(AMS) is completed, start writing to CCCD of Entity Update(AMS).
4. Writing to CCCD of Entity Update(AMS) is completed.

Code 2. Enable Notification

3.4.6 Remote Command characteristic features

3.4.6.1 Receive supported remote command list from MS (iOS device)

When MR (RA4W1) receives a GATT notification containing a supported remote command list from Remote Command characteristic in MS (iOS device). *BLE_AMSC_EVENT_RC_HDL_VAL_NTF* event will be notified to *AMSc_cb* function. Following procedures will perform in the event handler.

1. Decode the supported command list from the GATT notification.
2. Store the supported command list.
3. Print received command list on the console.

```
static void AMSc_cb(uint16_t type, ble_status_t result, st_ble_servc_evt_data_t *p_data)
{
    switch(type)
    {
……
        case BLE_AMSC_EVENT_RC_HDL_VAL_NTF:
        {
            R_BLE_CLI_Printf("receive BLE_AMSC_EVENT_RC_HDL_VAL_NTF\n");

            R_BLE_CLI_Printf("\nEvent: Received Notification ………");

            st_ble_seq_data_t *rc_ntf_data = (st_ble_seq_data_t *)p_data->p_param;

            /* Clear supported command list in MR(RA4W1) */
            memset(gs_rc_supported_cmd_list, 0, sizeof(gs_rc_supported_cmd_list));

            /* Updates supported command list in MR(RA4W1) based on the received data */
            for(int i=0; i<rc_ntf_data->len; i++)
            {
                gs_rc_supported_cmd_list[rc_ntf_data->data[i]] = true;
            }

            /* Display the list in the console */
            AMSc_print_rc_supported_cmd_list();

            break;
       }
……
}
```

1. Decode the received notification as [st_ble_seq_data_t](variable length data).

2. Store the supported command list.

3. Print received command list on the console .

Code 3. Decode, store, and display supported command list

3.4.6.2 Control media playback of MS (iOS device) by sending a remote command

To control the media playback of MS (iOS device), send a remote command to the Remote Command characteristic of MS (iOS device) by using *“amsc send_rc”* command on the console. The command will call *R_BLE_AMSC_WriteRc* API to send a Write Request to the Remote Command characteristic of MS (iOS device).

BLE_AMSC_EVENT_RC_WRITE_RSP event will be notified to *AMSc_cb* function when the write request is successfully accepted to MS (iOS device). The event handler does not implement since there is no specific procedure.

```
static void cmd_amsc_send_rc(int argc, char *argv[])
{
    uint16_t conn_hdl;
    uint8_t rc_data = 0;
    const st_ble_seq_data_t seq_rc_data =
    {
        .len = AMS_CMD_SEND_RC_LEN,
        .data = &rc_data,
    };
………

    conn_hdl = (uint16_t)strtol(argv[1], &str_check, 0);
    if(*str_check != '\0')
    {
        R_BLE_CLI_Printf("amsc %s: wrong parameter\n", argv[0]);
        return;
    }

    rc_data = (uint8_t)strtol(argv[2], &str_check, 0);
    if(*str_check != '\0')
    {
        R_BLE_CLI_Printf("amsc %s: wrong parameter\n", argv[0]);
        return;
    }

    if(BLE_AMSC_RC_REMOTECOMMANDID_BOOKMARKTRACK < rc_data)
    {
        R_BLE_CLI_Printf("amsc %s: undefined remote command\n", argv[0]);
        return;
    }
………

    result = R_BLE_AMSC_WriteRc(conn_hdl, &seq_rc_data);
………
}
```

Convert command argument to number.

Convert command argument to number.

Check argument.

Send Write Request of Remote Command characteristic.

Code 4. Decode, store, and display supported command list

3.4.7 Entity Update characteristic features

3.4.7.1 Send entity update command to subscribe attribute(s)

To send entity update command to MS (iOS device), the user can use “*amsc disp_eu*” command on the console. The command will call *R_BLE_AMSC_WriteEu* API to send a Write Request to Entity Update characteristic of MS (iOS device). *BLE_AMSC_EVENT_EU_WRITE_RSP* event will be notified to *AMSc_cb* function when the Write Request is successfully accepted to MS (iOS device). The event handler does not implement since there is no specific procedure.

```
static void cmd_amsc_disp_eu(int argc, char *argv[])
{
………
    for(int i=0; i+2<argc; i++)
    {
        eu_data[i] = (uint8_t)strtol(argv[i+2], &str_check, 0);
        if(*str_check != '\0')
        {
            R_BLE_CLI_Printf("amsc %s: wrong parameter\n", argv[0]);
            return;
        }

        if((0 != i) && (ams_attributes_num[eu_data[0]] <= eu_data[i]))
        {
            R_BLE_CLI_Printf("amsc %s: undefined AttributeID\n", argv[0]);
            return;
        }
    }

    if(BLE_AMSC_EUEA_ENTITYID_TRACK < eu_data[0])
    {
        R_BLE_CLI_Printf("amsc %s: undefined EntityID\n", argv[0]);
        return;
    }

    if(ams_attributes_num[eu_data[0]] < argc-3)
    {
        R_BLE_CLI_Printf("amsc %s: too many attributes\n", argv[0]);
        return;
    }
   ………
<Continue next page>
```

Convert argument as long value and store it.

Check argument

Check argument

Check argument

Check argument.

Code 5. Send Write Request to Entity Update characteristic (1/2)

```
<From previous page>
………
    for(int i = 1; i<seq_eu_data.len;i++)
    {
        R_BLE_CLI_Printf("Attribute ID : 0x%02x %s\n", eu_data[i], ………);
    }
    R_BLE_CLI_Printf("\nEnd Of List\n\n");

    result = R_BLE_AMSC_WriteEu(conn_hdl, &seq_eu_data);

    if(BLE_SUCCESS != result)
    {
        R_BLE_CLI_Printf("amsc %s: Write failed.\n Result: 0x%2X\n", argv[0], result);
    }
}
```

Display the data to be sent.

Send write request to MS.

Code 6. Send Write Request to Entity Update characteristic (2/2)

3.4.7.2 Receive an attribute value from MS (iOS device) as GATT notification

When a subscribed attribute has changed, MS (iOS device) will send the attribute value as GATT notification from the Entity Update characteristic. *BLE_AMCS_EVENT_EU_HDL_VAL_NTF* event will be notified to *AMS_cb* function when MR (RA4W1) has received the notification from MR (iOS device). Implementation of the event handler is shown following.

```
static void AMSc_cb(uint16_t type, ble_status_t result, st_ble_servc_evt_data_t *p_data)
{
……
       case BLE_AMSC_EVENT_EU_HDL_VAL_NTF:
       {
           R_BLE_CLI_Printf("receive BLE_AMSC_EVENT_EU_HDL_VAL_NTF\n");

           st_ble_seq_data_t *eu_data = (st_ble_seq_data_t*) p_data->p_param;
           R_BLE_CLI_Printf("Entity ID : 0x%02x %s\n", eu_data->data[0],………);
           R_BLE_CLI_Printf("Attribute ID : 0x%02x %s\n", eu_data->data[1],………);
           R_BLE_CLI_Printf("EntityUpdateFlag : 0x%02x %s\n", eu_data->data[2],………);
           R_BLE_CLI_Printf("Value : ");
           for(int i=3; i<eu_data->len; i++)
           {
               R_BLE_CLI_Printf("%c", eu_data->data[i]);
           }
           R_BLE_CLI_Printf("\n\n");
           break;
       }
……
}
```

Print received attribute

Code 7. Displaying received attribute from Entity Update characteristic

3.4.8 Entity Attribute characteristic features

The maximum length of attribute value from MS (iOS device) depends on MTU size. When the length of the attribute value is longer than MTU size, the attribute value will be truncated. In such a case, the user can get the full value of the attribute by using *"amsc get_ea"* command. The command will call *R_BLE_AMSC_WriteEa* API to send a Write Request to the Entity Attribute characteristic of MS (iOS device). *BLE_AMSC_EVENT_EA_WRITE_RSP* event will be notified to *AMSc_cb* function when the write request is successfully accepted to MS (iOS device). The write response event indicates that MS (iOS device) is ready to reply the full value of the truncated attribute. In this sample application, MR (RA4W1) will send a Read Request to MS (iOS device) in *BLE_AMSC_EVENT_EA_WRITE_RSP* event handler to get the full value. Implementation of these procedures is shown following.

```
static void cmd_amsc_get_ea(int argc, char *argv[])
{
………
    if (argc != 4)
    {
        R_BLE_CLI_Printf("amsc %s: unrecognized operands\n", argv[0]);
        return;
    }

    conn_hdl = (uint16_t)strtol(argv[1], &str_check, 0);
    for(int i =0; i+2<argc; i++)
    {
        ea_data[i] = (uint8_t)strtol(argv[i+2], &str_check, 0);
        if(*str_check != '\0')
        {
            R_BLE_CLI_Printf("amsc %s: wrong parameter\n", argv[0]);
            return;
        }

        if((0 != i) && (ams_attributes_num[ea_data[0]] <= ea_data[i]))
        {
            R_BLE_CLI_Printf("amsc %s: undefined AttributeID\n", argv[0]);
            return;
        }
    }

    if(BLE_AMSC_EUEA_ENTITYID_TRACK < ea_data[0])
    {
        R_BLE_CLI_Printf("amsc %s: undefined EntityID\n", argv[0]);
        return;
    }
………
    result = R_BLE_AMSC_WriteEa(conn_hdl, &seq_ea_data);
}
```

Check argument(s).

Convert argument string to long value.

Check argument(s).

Check argument(s).

Check argument(s).

Send write request to MS.

Code 8. Send the Entity Attribute command

```
static void AMSc_cb(uint16_t type, ble_status_t result, st_ble_servc_evt_data_t *p_data)
{
…….
       case BLE_AMSC_EVENT_EA_WRITE_RSP :
       {
           R_BLE_CLI_Printf("receive BLE_AMSC_EVENT_EA_WRITE_RSP\n");

           if(BLE_SUCCESS == result)
           {
               /*Read Entity Attribute*/
               R_BLE_AMSC_ReadEa(p_data->conn_hdl);
           }
           break;
       }

       case BLE_AMSC_EVENT_EA_READ_RSP :
       {
           R_BLE_CLI_Printf("receive BLE_AMSC_EVENT_EA_READ_RSP\n");

           if(BLE_SUCCESS == result)
           {
               /*Display Entity Attribute*/
               st_ble_seq_data_t *ea_read_data = (st_ble_seq_data_t *)p_data->p_param;
               R_BLE_CLI_Printf("\nEvent: Data Read from Entity Attribute characteristic
                                   \n\n");
               R_BLE_CLI_Printf("AMS: Following Read Received \n\n");
               R_BLE_CLI_Printf("Value: \"");
               for(int i =0;i<ea_read_data->len; i++)
               {
                   R_BLE_CLI_Printf("%c",ea_read_data->data[i]);
               }
               R_BLE_CLI_Printf("\"\n\nEnd Of List\n");
           }
           break;
       }
……
}
```

Send read request to MS.

Display received attribute value

Code 9. Entity Attribute event handing

3.4.9 AMS Error handling

BLE_GATTC_EVENT_ERROR_RSP event will be notified to *gattc_cb* when a GATT communication error has happened. This sample application prints which characteristic notifies the error response and its error code. Please refer to *Apple Media Service Reference* for detail of the error code.

```
void gattc_cb(uint16_t type, ble_status_t result, st_ble_gattc_evt_data_t *p_data)
{
       case BLE_GATTC_EVENT_ERROR_RSP:
       {
………
           /* Error response for Remote Command */
           st_ble_AMSc_rc_attr_hdl_t rc_hdl;
           R_BLE_AMSC_GetRcAttrHdl(&g_conn_bd_addr, &rc_hdl);
           if(err_rsp_data->attr_hdl == rc_hdl.range.start_hdl+1)
           {
              R_BLE_CLI_Printf("AMS : %s Error in RC 0x%04X \n",
                             ams_error_code_name[err_rsp_data->rsp_code & 0x0FU],
                             err_rsp_data->rsp_code);
           }

           /* Error response for Entity Update */
           st_ble_AMSc_eu_attr_hdl_t eu_hdl;
           R_BLE_AMSC_GetEuAttrHdl(&g_conn_bd_addr, &eu_hdl);
           if(err_rsp_data->attr_hdl == eu_hdl.range.start_hdl+1)
           {
               R_BLE_CLI_Printf("AMS : %s Error in EU 0x%04X \n",
                               ams_error_code_name[err_rsp_data->rsp_code & 0x0FU],
                               err_rsp_data->rsp_code);
           }

           /* Error response for Entity Attribute */
           st_ble_AMSc_ea_attr_hdl_t ea_hdl;
           R_BLE_AMSC_GetEaAttrHdl(&g_conn_bd_addr, &ea_hdl);
           if(err_rsp_data->attr_hdl == ea_hdl.range.start_hdl+1)
           {
               R_BLE_CLI_Printf("AMS : %s Error in EA 0x%04X \n",
                               ams_error_code_name[err_rsp_data->rsp_code & 0x0FU],
                               err_rsp_data->rsp_code);
           }
       } break;
/* End user code. Do not edit comment generated here */
    }
}
```

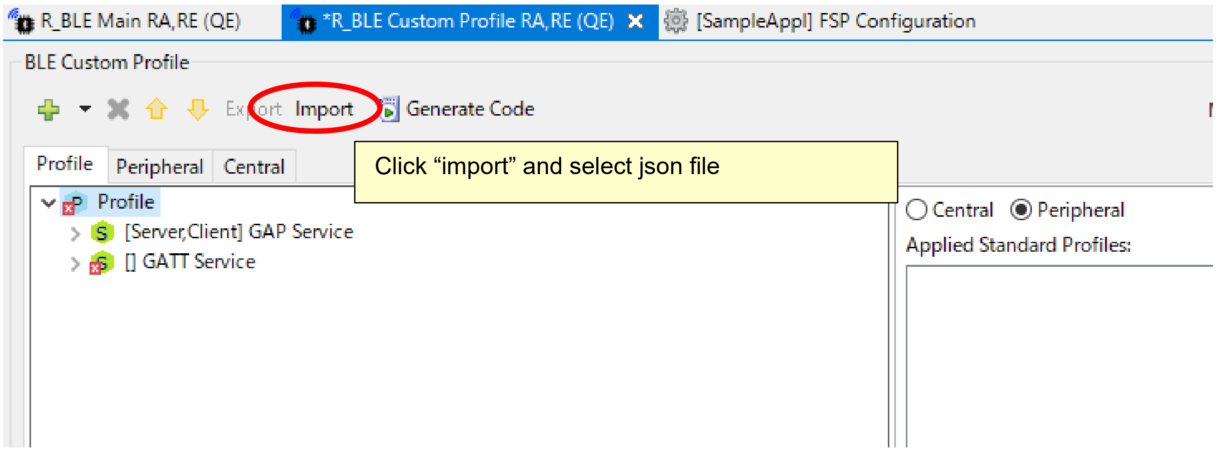
Code 10. Error handing

3.5 FreeRTOS program

Implementation is same as *RA4W1 Group Apple Notification Center Service Sample Program* (R01AN6071). Please refer to it.

3.6 Development using QE for BLE

The project of this application is developed based on programs generated using QE for BLE's custom service generation feature. When adding other services to the project, QE for BLE makes development easier. For instructions about usage of QE for BLE, please refer [*Bluetooth Low Energy Profile Developer's Guide (R01AN5428)*]. To create AMS client role, user can import *./src/ams_client.json* using import function of QE for BLE. This sample application also attached *./src/ancs_client.json* for your convenience. To import these json file to your own project, it is necessary to click *import* and select json files in QE for BLE.



Click “import” and select json file

Figure 10. QE for BLE import button

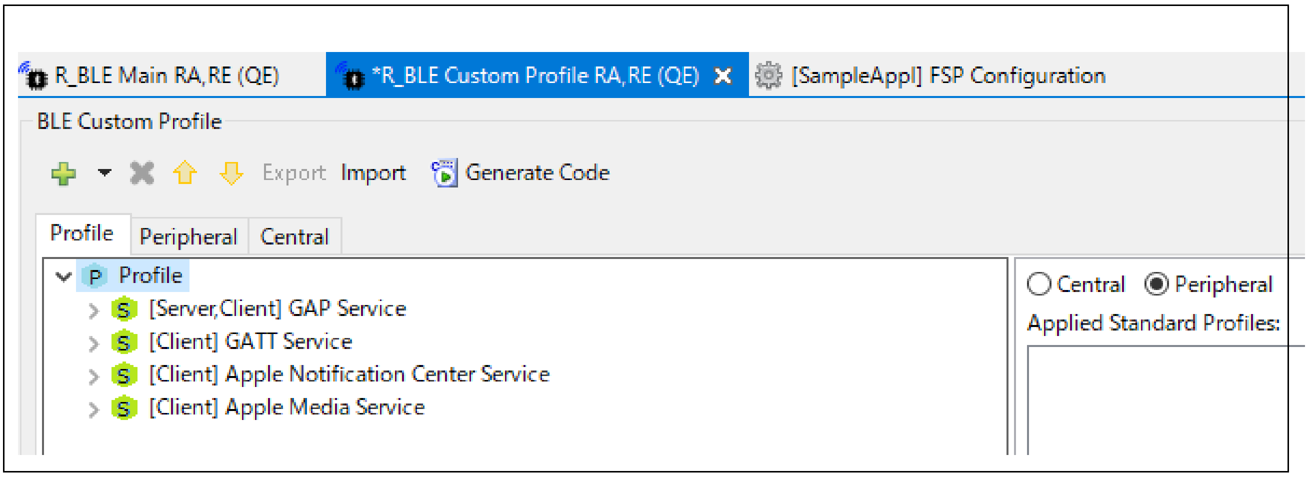

R_BLE Main RA,RE (QE)
*R_BLE Custom Profile RA,RE (QE)
[SampleAppl] FSP Configuration
BLE Custom Profile
Export
Import
Generate Code
Profile
Peripheral
Central
Profile
[Server,Client] GAP Service
[Client] GATT Service
[Client] Apple Notification Center Service
[Client] Apple Media Service
Central
Peripheral
Applied Standard Profiles:

Figure 11. QE for BLE after import json file

After code generation, please add following enumeration declaration to r_ble_ANCSc.h and r_ble_AMSc.h. For more information about data format or value, please refer to *Apple Notification Center Service (ANCS) Specification* and *Apple Media Service Reference*.

```
/*****************************************************************************//**
* @brief Control Point/Data Source Command ID enumeration.
********************************************************************************/
typedef enum {
BLE_ANCSC_CPDS_COMMANDID_GETNOTIFICATIONATTRIBUTE = 0, /**< Get notification attribute Command */
BLE_ANCSC_CPDS_COMMANDID_GETAPPATTRIBUTE = 1, /**< Get app attribute Command */
BLE_ANCSC_CPDS_COMMANDID_PERFORMNOTIFICATIONACTION = 2, /**< Perform notification action Command */
} e_ble_ancsc_cpds_commandid_t;
/*****************************************************************************//**
* @brief Control Point/Data Source Notification Attribute ID enumeration.
********************************************************************************/
typedef enum {
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_APPIDENTIFIER = 0, /**< Notification Attribute ID: App Identifier */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_TITLE = 1, /**< Notification Attribute ID: Title */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_SUBTITLE = 2, /**< Notification Attribute ID: Sub title */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_MESSAGE = 3, /**< Notification Attribute ID: Message */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_MESSAGESIZE = 4, /**< Notification Attribute ID: Message size */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_DATE = 5, /**< Notification Attribute ID: Date */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_POSITIVEACTIONLABEL = 6, /**< Notification Attribute ID: Positive action label */
BLE_ANCSC_CPDS_NOTIFICATIONATTRIBUTEID_NEGATIVEACTIONLABEL = 7, /**< Notification Attribute ID: Negative action label
*/
} e_ble_ancsc_cpds_notificationattributeid_t;
/*****************************************************************************//**
* @brief Control Point/Data Source App Attribute ID enumeration.
********************************************************************************/
typedef enum {
BLE_ANCSC_CPDS_APPATTRIBUTEID_DISPLAYNAME = 0, /**< App Attribute ID: Display name */
} e_ble_ancsc_cpds_appattributeid_t;
/*****************************************************************************//**
* @brief Control Point/Data Source Action ID enumeration.
********************************************************************************/
typedef enum {
BLE_ANCSC_CPDS_ACTIONID_POSITIVE = 0, /**< Positive action */
BLE_ANCSC_CPDS_ACTIONID_NEGATIVE = 1, /**< Negative action */
} e_ble_ancsc_cpds_actionid_t;
```

Code 11. r_ble_ANCSc.h

```
/*****************************************************************************//**
 * @brief Remote Command ID enumeration.
*******************************************************************************/
typedef enum {
    BLE_AMSC_RC_REMOTECOMMANDID_PLAY                   = 0, /**< RemoteCommand ID: Play */
    BLE_AMSC_RC_REMOTECOMMANDID_PAUSE                  = 1, /**< RemoteCommand ID: Pause */
    BLE_AMSC_RC_REMOTECOMMANDID_TOGGLEPLAYPAUSE        = 2, /**< RemoteCommand ID: Toggle Play Pause */
    BLE_AMSC_RC_REMOTECOMMANDID_NEXTTRACK              = 3, /**< RemoteCommand ID: Next Track */
    BLE_AMSC_RC_REMOTECOMMANDID_PREVIOUSTRACK          = 4, /**< RemoteCommand ID: Previous Track */
    BLE_AMSC_RC_REMOTECOMMANDID_VOLUMEUP               = 5, /**< RemoteCommand ID: Volume Up */
    BLE_AMSC_RC_REMOTECOMMANDID_VOLUMEDOWN             = 6, /**< RemoteCommand ID: Volume Down */
    BLE_AMSC_RC_REMOTECOMMANDID_ADVANCEREPEATMODE      = 7, /**< RemoteCommand ID: Advance Repeat Mode */
    BLE_AMSC_RC_REMOTECOMMANDID_ADVANCESHUFFLEMODE     = 8, /**< RemoteCommand ID: Advance Shuffle Mode */
    BLE_AMSC_RC_REMOTECOMMANDID_SKIPFORWARD            = 9, /**< RemoteCommand ID: Skip Forward */
    BLE_AMSC_RC_REMOTECOMMANDID_SKIPBACKWARD           = 10, /**< RemoteCommand ID: Skip Backward */
    BLE_AMSC_RC_REMOTECOMMANDID_LIKETRACK              = 11, /**< RemoteCommand ID: Like Track */
    BLE_AMSC_RC_REMOTECOMMANDID_DISLIKETRACK           = 12, /**< RemoteCommand ID: Dislike Track */
    BLE_AMSC_RC_REMOTECOMMANDID_BOOKMARKTRACK          = 13, /**< RemoteCommand ID: Bookmark Track */
} e_ble_AMSc_rc_remotecommandid_t;

/*****************************************************************************//**
 * @brief Entity Update/Entity Attribute Entity ID enumeration.
*******************************************************************************/
typedef enum {
    BLE_AMSC_EUEA_ENTITYID_PLAYER = 0,
    BLE_AMSC_EUEA_ENTITYID_QUEUE = 1,
    BLE_AMSC_EUEA_ENTITYID_TRACK = 2,
} e_ble_amsc_euea_entityid_t;

<Continue to next page>
```

Code 12. r_ble_AMSc.h (1/3)

<From previous page>

```
/****************************************************************************//**
 * @brief Entity Update/Entity Attribute Player Attribute ID enumeration.
********************************************************************************/
typedef enum {
    BLE_AMSC_EUEA_PLAYERATTRIBUTEID_NAME = 0,
    BLE_AMSC_EUEA_PLAYERATTRIBUTEID_PLAYBACKINFO = 1,
    BLE_AMSC_EUEA_PLAYERATTRIBUTEID_VOLUME = 2,
} e_ble_AMSc_euea_playerattributeid_t;

/****************************************************************************//**
 * @brief Entity Update/Entity Attribute Queue Attribute ID enumeration.
********************************************************************************/
typedef enum {
    BLE_AMSC_EUEA_QUEUEATTRIBUTEID_INDEX = 0,
    BLE_AMSC_EUEA_QUEUEATTRIBUTEID_COUNT = 1,
    BLE_AMSC_EUEA_QUEUEATTRIBUTEID_SHUFFLEMODE = 2,
    BLE_AMSC_EUEA_QUEUEATTRIBUTEID_REPEATMODE = 3,
} e_ble_AMSc_euea_queueattributeid_t;

/****************************************************************************//**
 * @brief Entity Update/Entity Attribute Track Attribute ID enumeration.
********************************************************************************/
typedef enum {
    BLE_AMSC_EUEA_TRACKATTRIBUTEID_AIRTIST = 0,
    BLE_AMSC_EUEA_TRACKATTRIBUTEID_ALBUM = 1,
    BLE_AMSC_EUEA_TRACKATTRIBUTEID_TITLE = 2,
    BLE_AMSC_EUEA_TRACKATTRIBUTEID_DURATION = 3,
} e_ble_AMSc_euea_trackattributeid_t;
```

Code 13. r_ble_AMSc.h (2/3)

```
<From previous page>
/****************************************************************************//**
 * @brief Entity Update/Entity Attribute Shuffle Mode enumeration.
********************************************************************************/
typedef enum {
    BLE_AMSC_EUEA_SHUFFLEMODE_OFF = 0,
    BLE_AMSC_EUEA_SHUFFLEMODE_ONE = 1,
    BLE_AMSC_EUEA_SHUFFLEMODE_ALL = 2,
} e_ble_AMSc_euea_shufflemode_t;

/****************************************************************************//**
 * @brief Entity Update/Entity Attribute Repeat Mode enumeration.
********************************************************************************/
typedef enum {
    BLE_AMSC_EUEA_REPEATMODE_OFF = 0,
    BLE_AMSC_EUEA_REPEATMODE_ONE = 1,
    BLE_AMSC_EUEA_REPEATMODE_ALL = 2,
} e_ble_AMSc_euea_repeatmode_t;

/****************************************************************************//**
 * @brief Entity Update Flag enumeration.
********************************************************************************/
typedef enum {
    BLE_AMSC_EU_ENTITYUPDATEFLAG_TRUNCATED = 0,
} e_ble_AMSc_eu_entityupdateflag_t;
```

Code 14. r_ble_AMSc.h (3/3)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.20.21	-	First edition issued.
1.01	Nov.11.22	5	Updated for FSP v4.1.0

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.