RW BLE Find Me Profile Interface Specification

Interface Specification
RW-BLE-PRF-FMP-IS
Version 8.00
2015-07-29

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.1	March 11th 2011	Initial release	CI
1.0	December 15th 2011	Interface update	CI
2.0	August 14th 2012	API Update	LT
3.0	December 3rd 2012	Client Multi-Instance API	LT
7.00	November 11th 2014	Updated for BLE 4.1	FBE
8.00	July 29th 2015	Updated for BLE 4.2	CM

Table of Contents

1 Overview

1.1 Document Overview

This document describes the non-standard interface of the RW BLE Find Me Profile (FMP) implementation. Along this document, the interface messages will be referred to as API messages for the profile block(s).

Their description will include their utility and reason for implementation for a better understanding of the user and the developer that may one day need to interface them from a higher application.

1.2 Protocol Overview

The Bluetooth Low Energy Find Me profile enables the user to locate a device using another device connected to it. Within the profile, two roles can be supported: **Locator** and **Target**. The Locator must support the GAP Central Role and the Target, the GAP Peripheral role. The profile requires a connection to be established between the two devices for its functionality.

The functionality of a profile requires the presence of certain services and attributes on one of the two devices, which the other device can manipulate. In this case, the Find Me Target device must have an instance of the Immediate Alert Service (IAS) in its attribute database. The Find Me Locator (FINDL) will discover this service and its Alert Level Characteristic, which it may then write to one of the three possible alert levels (None/Mild/High) to cause the Target(FINDT) device to alert or stop alerting once it has been located.

The various documents edited by the Bluetooth SIG PUID Working group present different use cases for this profile, their GATT, GAP and security, mandatory and optional requirements. The FMP profile and IAS Service specifications have been adopted by the Bluetooth SIG on June 21st 2011 ([1], [2]). Their related Test Specifications have been released at the same time and are referenced in [3], [4].

The profile is implemented in the RW-BLE software stack as two tasks, one for each role. Each task has an API decided after the study of the profile specifications and test specifications, and it is considered to be minimalistic and designed for a future application which would combine the profile functionality with the device connectivity and security procedures.

1.3 Firmware Implementation Overview

Basically, if a device needs only be FMP Target, the firmware should be compiled with this role only, and inversely for the Locator role. The role enables the part of the DB, which, important to know, will be hidden by the Target until the Target role is enabled post-connection establishment.

The Applications which will control the roles on end-products are responsible with creating the connection between the devices, using suggested advertising intervals and data, connection intervals, security levels, etc. The Profile implementation allows modulating the behavior depending on the final needs. Profile role enabling should be immediate after connection creation in order to allow correct profile behavior with the peer device.

The Alert levels will also be interpreted by the application which receives an indication of their changed value and will control physical device components for starting/stopping a certain level of alert (LEDs, buzzer, etc.). For example, as the Firmware is tested now, we can only see the indication of the alert level in the Test application, the final implementation of this Profile within a real use case should be considered in a final product scenario only.

2 Find Me Target

This role is meant to be activated on the device that needs to be found. It implies it is a GAP Peripheral. The FW task for this role will interpret the values written in the Immediate Alert Service Alert Level characteristic by the Locator peer. Please refer to “findt_task.h” for implementation of this API.

Find Me Target task is a mono-instantiated task; it means that connection index is **not** present into task index.

2.1 Initialization / Database Creation

During the initialization phase of the device, to use the Find Me Target task, the FINDT task has to be allocated and corresponding attribute database initialized, using GAPM API. Application has to send GAPM_PROFILE_TASK_ADD_CMD [6] with specific device required security level and following parameters.

Parameters: None

2.2 FINDT_ALERT_IND

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection index of the peer device which has modify alert level
uint8_t	alert_lvl	Alert level that has been set in the IAS Alert Level Characteristic.

Response: None

Description: This API message is used by the Target role to inform the Application of a valid alert level written by the peer in the IAS Alert Level Characteristic. The message is created and sent after reception and check of a GATT_WRITE_CMD_IND forwarded by TASK_SVC. The connection handle and application task ID stored in the Target environment are used for the creation of the kernel message. Only if a valid level has been written by peer (“No Alert” = 0, “Mild Alert” = 1, “High Alert” = 2) will this message be sent to the application. In any other case the attribute value may be updated in the database, but it will not trigger profile functionality.

The Application alone is responsible for actually triggering/stopping a noticeable visual/audio alert on the device upon reception of this indication.

3 Find Me Locator

This role is meant to be activated on the device that will locate the Target. It implies it is a GAP Central. The FW task for this role will discover the Immediate Alert Service present on the peer Server, after establishing connection, and will allow writing different alert levels to the Alert Level characteristic in the IAS. Please refer to "findl_task.h" for implementation of this API.

Important Note: The TASK_FINDL task is multi-instantiated, one instance is created for each connection for which the profile will be enabled and each of these instances will have a different task ID. To communicate with the peer device, the corresponding connection index has to be used to calculate the FINDL task instance

The term TASK_FINDL_IDX will be used in the rest of the document to refer to any instance of the Find Me Profile Locator Role Task. The term TASK_FINDL will refer to the first instance of this task.

3.1 Initialization

During the initialization phase of the device, to use the Find Me Locator task, the FINDL task has to be allocated using GAPM API. Application has to send GAPM_PROFILE_TASK_ADD_CMD [6].

3.2 FINDL_ENABLE_REQ

Source: TASK_APP

Destination: TASK_FINDL_IDX

Parameters:

Type	Parameters	Description
uint8_t	con_type	Connection type: 1st discovery or normal connection.
struct ias_content	ias	Save IAS information (see Table 4.2 – HTS Content (struct htpc_hts_content))

Response: FINDL_ENABLE_RSP

Description: This API message is used for enabling the Locator role of the Find Me profile. The Application sends it, and it contains the connection handle for the connection this profile is activated, the connection type and the previously saved discovered IAS details on peer. The Task for this profile role will go from IDLE state to CONNECTED state.

The connection type may be *0 = Connection for discovery* or *1 = Normal connection*. This difference has been made and Application would handle it in order to not discover the IAS on the Target at every connection, but do it only once and keep the discovered details in the Locator device between connections.

***ATTENTION**: Normally information about the peer should not be kept from one connection to the next if they have not bonded!*

If it is a discovery type connection, the *ias* parameter is useless, it will be filled with 0's. Otherwise it will contain pertinent data which will be kept in the Locator environment while enabled. It allows for the Application to not be aware of attribute details. For a normal connection, the response to this request is sent right away after saving the *ias* content in the environment. For a discovery connection, discovery of the peer IAS is started and the response will be sent at the end of the discovery with the discovered attribute details.

3.3 FINDL_ENABLE_RSP

Source: TASK_FINDL_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status code (see [5])
struct ias_content	ias	Discovery result (see Table 4.2 – HTS Content (struct htpc_hts_content))

Description: This API message is used by the Locator to either send the discovery results of IAS on Target and confirm enabling of the Locator role, or to simply confirm enabling of Locator role if it is a normal connection and the IAS details are already known. An error may have also occurred and is signaled.

3.4 FINDL_SET_ALERT_REQ

Source: TASK_FINDL_IDX

Destination: TASK_FINDL

Parameters:

Type	Parameters	Description
uint8_t	alert_lvl	Alert level to set in Target. Three levels are defined: FINDL_ALERT_NONE FINDL_ALERT_MILD FINDL_ALERT_HIGH

Response: FINDL_SET_ALERT_RSP

Description: This API message is used by the application to trigger/stop and alert on the peer Target device. The Locator role environment contains the attribute handle for the Alert Level Characteristic in the IAS of the Target peer device. This way, a correct request to write this attribute to the level requested by the application can be sent. Since the Alert Level Characteristic in IAS can only be written using a Write No Response ATT Request.

3.5 FINDL_SET_ALERT_RSP

Source: TASK_FINDL_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status code (see [5])

Description: Inform Application that Alert has been written on peer device.

4 Miscellaneous

Type	Parameters	Description
struct prf_svc	svc	Structure containing the start handle and the end handle of the service /// start handle uint16_t shdl; /// end handle uint16_t ehdl;
struct prf_char_inf	alert_lvl_char	Structure containing handle and properties of the alert level characteristic

Table 4.1 – HTS Content (struct htpc_hts_content)

5 Abbreviations

Abbreviation	Original Terminology
API	Application Programming Interface
ATT	Attribute
BLE	Bluetooth Low Energy
DB	Database
FINDL	Find Me Locator
FINDT	Find Me Target
FMP	Find Me Profile (official acronym)
FW	Firmware
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IAS	Immediate Alert Service
ID	Identifier
IS	Interface Specification
PRF	Profile
RW	RivieraWaves
SVC	Service

References

[1]	**Title**	Find Me Profile		
	Reference	FMP_SPEC_V10		
	Version	V10r00	**Date**	21/06/2011
	Source	Bluetooth SIG - PUID Working Group		

[2]	**Title**	Immediate Alert Service		
	Reference	IAS_SPEC_V10		
	Version	V10r00	**Date**	21/06/2011
	Source	Bluetooth SIG - PUID Working Group		

[3]	**Title**	Find Me Profile Test Specification 1.0		
	Reference	FMP.TS.1.0.0		
	Version	1.0.0	**Date**	26/06/2011
	Source	Bluetooth SIG		

[4]	**Title**	Immediate Alert Service Test Specification 1.0		
	Reference	IAS.TS.1.0.0		
	Version	1.0.0	**Date**	27/06/2011
	Source	Bluetooth SIG		

[5]	**Title**	RW BLE Host Error Code Interface Specification		
	Reference	RW-BLE-HOST-ERR-CODE-IS		
	Version	7.00	**Date**	2014-06-30
	Source	RivieraWaves SAS		

[6]	**Title**	GAP Interface Specification		
	Reference	RW-BLE-GAP-IS		
	Version	7.00	**Date**	2014-06-30
	Source	RivieraWaves SAS		