RW BLE Proximity Profile Interface Specification

Interface Specification

RW-BLE-PRF-PXP-IS

Version 9.01

2018-11-12

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.1	2010-11-03	Initial release	CI
1.0	2011-12-15	Updated API	CI
2.0	2012-08-14	Updated API	LT
3.0	2012-12-03	Client Multi-Instance API	LT
7.00	2015-11-28	Updated for BLE 4.1	FBE
8.00	2015-07-29	Updated for BLE 4.2	CM
9.0	2017-03-09	Updated for BLE 5	LT
9.01	2018-11-12	Corrected prox database config structure	KYAP

Table of Contents

1 Overview

1.1 Document Overview

This document describes the non-standard interface of the RW BLE Proximity Profile 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 Proximity profile enables the user to monitor proximity/link loss between two devices. Within the profile, two roles can be supported: ***Monitor*** and ***Reporter***. The Monitor must support the GAP Central Role and the Reporter, 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 Proximity Reporter device must have an instance of the Link Loss Service (LLS), and may also have the Immediate Alert Service (IAS) and Tx Power Service (TPS) in its attribute database. The two last ones must be used together, if one is missing, the other one should be ignored. The Proximity Monitor will discover these services and their Characteristics: the Alert Level Characteristic in LLS, the Alert Level Characteristic in IAS and the Tx Power Level Characteristic in TPS.

The LLS allows the user to set an alert level in the Reporter, which will be used by the reporter to alert in the corresponding way if the link is lost. The disconnection must not come voluntarily from one of the two devices in order to trigger the alert.

The IAS allows the user to set an immediate alert level based on path loss computation using the read Tx Power Level and RSSI monitored on received packets. According to the alert level set in IAS, the Reporter will start alerting immediately.

The TPS allows the user to read the Tx Power Level for the physical layer. The value is used by the Monitor to continuously evaluate path loss during the connection, and decide to trigger/stop an alert based on path loss going over/under a set threshold in the Monitor application.

The security requirements by this profile are enforced through the Attribute properties, and by the Application: the Application on the Reporter device should start security procedure prior to enabling the Reporter role, just as the Application on the Monitor device should enable the Monitor role after the security procedure is over. The user normally has knowledge of the security requirements of both devices so confusion is hard to take place.

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 PXP profile and IAS, LLS, TPS Service specifications have been adopted by the Bluetooth SIG on June 21st 2011 ([1], [2], [3], [4]). Their related Test Specifications have been released at the same time and are referenced in [5], [6], [7], [8].

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 PXP Reporter, the firmware should be compiled with this role only, and inversely for the Monitor role. The role enables the part of the DB, which, important to know, will be hidden by the Reporter until the Reporter 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 Proximity Reporter

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 LLS and IAS Alert Level characteristics by the Monitor peer. Please refer to “proxr_task.h” for implementation of this API.

Proximity Reporter 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 Proximity Reporter task, the PROXR task has to be allocated and corresponding attribute database initialized, using GAPM API. Application has to send GAPM_PROFILE_TASK_ADD_CMD [10] with specific device required security level and following parameters.

Parameters:

Type	Parameters	Description
uint16_t	features	Indicate if optional features are supported or not

The features parameter shall be used to indicate if IAS and TXPS are supported by the device (0x01) or not (0x00). The proximity profile specification indicates that both of these services are optional but have to be present together into the database.

Note: The Proximity reporter services are allocated in a contiguous handle range to simplify services allocation.

2.2 PROXR_ALERT_IND

Source: TASK_PROXR

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 according to which the Device must start alerting.
uint8_t	char_code	Char Code - Indicate if LLS or IAS (PROXR_IAS_CHAR or PROXR_LLS_CHAR)

Description: This API message is used by the Reporter role to request the Application to start the alert on the device considering the indicated alert level. The message may be created and sent on two conditions:

- The IAS alert level characteristic has been written to a valid value, in which case *alert_lvl* will be set to the IAS alert level value.
- A disconnection with a reason other than the normal local/remote link terminations has been received, in which case *alert_lvl* will be set to the LLS alert level value.

The connection handle and application task ID stored in the Target environment are used for the creation of the kernel message.

The Application actions following reception of this indication are strictly implementation specific (it may try to reconnect to the peer and stop alert upon that, or timeout the alert after a certain time, please see the specification)

3 Proximity Monitor

This role is meant to be activated on the device that will monitor the connection with the Reporter device. It implies it is a GAP Central. The FW task for this role will discover the LLS, IAS and TPS services present on the peer Server, after establishing connection, and will allow writing different alert levels to the Alert Level characteristic in the LLS or IAS, and also reading the Tx Power Level characteristic in TPS. Please refer to “proxm_task.h” for current API.

This task has 3 possible states: IDLE, CONNECTED, DISCOVERING.

Important Note: The TASK_PROXM 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 PROXM task instance

The term TASK_PROXM_IDX will be used in the rest of the document to refer to any instance of the Proximity Profile Monitor Role Task. The term TASK_PROXM will refer to the first instance of this task.

3.1 Initialization

During the initialization phase of the device, to use the Proximity Monitor task, the PROXM task has to be allocated using GAPM API. Application has to send GAPM_PROFILE_TASK_ADD_CMD [10].

3.2 PROXM_ENABLE_REQ

Source: TASK_APP

Destination: TASK_PROXM_IDX

Parameters:

Type	Parameters	Description
uint8_t	con_type	Connection type: 1st discovery or normal connection.
struct svc_content	lls	Link Loss Service saved information (see Table 4.2 – Service Content Structure (struct svc_content))
struct svc_content	ias	Immediate Alert Service saved information
struct svc_content	txps	TX Power Service saved information

Response: PROXM_ENABLE_RSP

Description: This API message is used for enabling the Monitor role of the Proximity 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 LLS, IAS and TPS details on peer.

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 attributes on the Reporter at every connection, but do it only once and keep the discovered details in the Monitor device between connections.

If it is a discovery type connection, the **lls , ias** and **tps** parameters are useless, they will be filled with 0's. Otherwise it will contain pertinent data which will be kept in the Monitor environment while enabled. It allows for the Application to not be aware of attribute details, and only give them to the profile role from a storage area where they are kept in between connections (NVDS...).

For a *normal* connection, the response to this request is sent right away after saving the lls, ias and tps content in the environment. For a *discovery* connection, discovery of the peer attributes is started and the response will be sent at the end of the discovery with the discovered attribute details. The discovery starts with the 3 services, then with the characteristics of the discovered services. If an error happens during discovery (mandatory conditions present in the

specification are not respected – missing services, mandatory characteristics...) it is signaled right away to the application setting an appropriate value in the *status* parameter of the response.

The Task for this profile role will go from IDLE state to CONNECTED state in a *normal* connection, and from IDLE to DISCOVERING state in a *discovering* connection.

3.3 PROXM_ENABLE_RSP

Source: TASK_PROXM_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status code (see [9])
struct svc_content	lls	Link Loss Service saved information (see Table 4.2 – Service Content Structure (struct svc_content))
struct svc_content	ias	Immediate Alert Service saved information
struct svc_content	txps	TX Power Service saved information

Description: This API message is used by the Monitor to either send the discovery results of LLS, IAS and TPS on Reporter and confirm enabling of the Monitor role, or to simply confirm enabling of the Monitor role if it is a normal connection and the LLS, IAS and TPS details are already known.

3.4 PROXM_RD_REQ

Source: TASK_APP

Destination: TASK_PROXM_IDX

Parameters:

Type	Parameters	Description
uint8_t	svc_code	0=LLS or 1=TXPS, code for the service in which the alert level should be read

Response: PROXM_RD_RSP

Description: This API message is used for reading the alert level in LLS Alert Level Characteristic or the TX Power level value of peer device service.

3.5 PROXM_RD_RSP

Source: TASK_PROXM_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	svc_code	0=LLS or 1=TXPS, code for the service in which the alert level should be read
uint8_t	status	Status code (see [9])
uint8_t	value	Tx Power level value or Link Loss alert configured value

Description: Response of the Read request of Link Loss alert value or TX Power value.

3.6 PROXM_WR_ALERT_LVL_REQ

Source: TASK_APP

Destination: TASK_PROXM_IDX

Parameters:

Type	Parameters	Description
uint8_t	svc_code	Simple 0/1 value to determine whether the Application wants the Alert characteristic in LLS(0) or IAS(1) to be written
uint8_t	lvl	Alert level to be set in the targeted characteristic.

Response: PROXM_WR_ALERT_LVL_RSP

Description: This API message is used by the application to set either a LLS Alert Level or an IAS Alert Level. Since these two service have characteristics of the same type (but not the same properties – LLS one is R&W, IAS one is Write no Response only), one API message for a very similar purpose was considered sufficient, and therefore a simple byte for differentiating whether the alert code should be set in LLS or IAS characteristic is used.

Upon reception of this request, the connection handle is checked, then the alert level to set in order to ensure a valid value, and then the *svc_code* for 0 or 1.. In case any of these checks fail, an PROXM_ERROR_IND message is sent to the Application with error code 0x01.

If the checks are successful, a write characteristic request is build for TASK_GATT, with the appropriate type (Simple Write or Write no response) and using the characteristic handle for the service indicated by *svc_code.* When the Write Response is received from TASK_GATT, it is sent to the Application for it to check the status.

3.7 PROXM_WR_ALERT_LVL_RSP

Source: TASK_PROXM_IDX

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Status code (see [9])
uint8_t	svc_code	Simple 0/1 value to determine whether the Application wants the Alert characteristic in LLS(0) or IAS(1) to be written

Description: Response of the write request of Link Loss or immediate alert value

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	charact	Structure containing handle and properties of the only characteristic of the service. /// characteristic handle uint16_t char_hdl; /// value handle uint16_t val_hdl; /// characteristic properties uint8_t prop;

Table 4.1 – Service Content Structure (struct svc_content)

5 Abbreviations

Abbreviation	Original Terminology
API	Application Programming Interface
ATT	Attribute
BLE	Bluetooth Low Energy
DB	Database
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IAS	Immediate Alert Service
LLS	Link Loss Service
PROXM	Proximity Monitor
PROXR	Proximity Reporter
PXP	Proximity Profile (official acronym)
RW	RivieraWaves
TPS	TX Power Service

References

[1]	**Title**	Proximity Profile		
	Reference	PXP_SPEC_V10		
	Version	V10r00	**Date**	June 21st 2011
	Source	Bluetooth SIG – PUID Working Group		

[2]	**Title**	Link Loss Service		
	Reference	LLS_SPEC_V10		
	Version	V10r00	**Date**	June 21st 2011
	Source	Bluetooth SIG – PUID Working Group		

[3]	**Title**	Immediate Alert Service		
	Reference	IAS_SPEC_V10		
	Version	V10r00	**Date**	June 21st 2011
	Source	Bluetooth SIG – PUID Working Group		

[4]	**Title**	TX Power Service		
	Reference	TPS_SPEC_V10		
	Version	V10r00	**Date**	June 21st 2011
	Source	Bluetooth SIG – PUID Working Group		

[5]	**Title**	Proximity Profile Test Specification		
	Reference	PXP.TS.1.0.0		
	Version	1.0.0	**Date**	June 27th 2011
	Source	Bluetooth SIG		

[6]	**Title**	Link Loss Service Test Specification		
	Reference	LLS.TS.1.0.0		
	Version	1.0.0	**Date**	June 26th 2011
	Source	Bluetooth SIG		

[7]	**Title**	Immediate Alert Service Test Specification 1.0		
	Reference	IAS.TS.1.0.0		
	Version	1.0.0	**Date**	June 27st 2011
	Source	Bluetooth SIG		

[8]	**Title**	TX Power Service Test Specification		
	Reference	TPS.TS.1.0.0		
	Version	1.0.0	**Date**	June 26th 2011
	Source	Bluetooth SIG		

[9]	**Title**	RW BLE Host Error Code Interface Specification		
	Reference	RW-BLE-HOST-ERR-CODE-IS		
	Version	9.0	**Date**	2017-03-09
	Source	RivieraWaves SAS		

[10]	**Title**	GAP Interface Specification		
	Reference	RW-BLE-GAP-IS		
	Version	9.0	**Date**	2017-03-09
	Source	RivieraWaves SAS		