RW BLE Health Thermometer Profile Interface Specification

Interface Specification

RW-BLE-PRF-HTP-IS

Version 9.0

2017-03-09

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.1	2011-03-03	Initial release	CI
1.0	2011-09-21	Post profile adoption changes	CI
1.1	2011-12-15	Update API	CI
2.0	2012-08-13	Database creation API	LT
3.0	2012-11-30	Client Multi-Instance API	LT
7.00	2014-11-28	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

Table of Contents

1 Overview

1.1 Document Overview

This document describes the non-standard interface of the RW BLE Health Thermometer 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 Health Thermometer profile enables the user to receive temperature measurements from a thermometer device and also configure it for different use cases. Within the profile, two roles can be supported: **Collector** and **Thermometer**. The Collector must support the GAP Central Role and the Thermometer, 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 Thermometer device must have one instance of the Health Thermometer Service (HTS) and one instance of Device Information Service (DIS) in its attribute database. The DIS shall be manage separately using the DISS task (please refer to the RW BLE Device Information Service Interface Specification).

The Health Thermometer Collector (HTPC) will discover these services and their characteristics, and it may then configure them to cause the Thermometer (HTPT) device to take measurements and indicate/notify them to the Collector.

The various documents edited by the Bluetooth SIG Medical Working group present different use cases for this profile, their GATT, GAP and security, mandatory and optional requirements. The HTP profile and HTS specifications have been adopted by the Bluetooth SIG on May 24th 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 HTP Thermometer, the firmware should be compiled with this role only, and inversely for the Collector role. The role enables the part of the DB, which, important to know, will be hidden by the Thermometer until the Thermometer 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.

2 Health Thermometer Profile Thermometer

This role is meant to be activated on the device that acts as Thermometer and sends measurement values to the Collector. It implies it is a GAP Peripheral. The FW task for this role will act following the configuration set by the Collector in the HTS characteristics. Please refer to "htpt_task.h" for implementation of this API.

This task only has two states, IDLE and BUSY.

Health thermometer service task is a mono-instantiated task; it means that connection index is **not** present into task index.

2.1 HTS Requirements

The following table gives an overview of the HTS structure :

Characteristic name	Requirement	Mandatory Properties	Optional properties	Security Permissions	Descriptors
Temperature Measurement	M	Indicate		None	Client Characteristic Configuration Req: M Prop: Read/Write
Temperature Type	O	Read		None	
Intermediate Temperature	O	Notify		None	Client Characteristic Configuration Req: M Prop: Read/Write
Measurement Interval	O	Read	Indicate Write	Writable with Authentication	Client Characteristic Configuration Req: C1 Prop: Read/Write
					Valid Range Req: C2 Prop: Read
M: Mandatory O: Optional C1: Mandatory if Measurement Interval characteristic supports sending of indications C2: Mandatory if Measurement Interval characteristic is writable					

2.2 DIS Requirements

The Health Thermometer profile requires the presence of three characteristics : ***Manufacturer Name String, Model Number String, System ID***. It is application's responsibility to add an instance of the DIS into the database by using the DISS_CREATE_DB_REQ API message (please see the RW BLE Device Information Service Interface Specification document).

2.3 Initialization / Database Creation

During the initialization phase of the device, to use the Health Thermometer Service task, the HTPT 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:

Type	Parameters	Description
uint8_t	features	Indicate if optional features are supported or not (see Table 3.1 – Database creation feature flags)
uint8_t	temp_type	Temperature Type Value
uint16_t	valid_range_min	Measurement Interval Valid Range - Minimal Value
uint16_t	valid_range_max	Measurement Interval Valid Range - Maximal Value
uint16_t	meas_intv	Measurement interval (latest known interval range)

The feature parameter shall be used to indicate if an optional feature is supported or not. Value of this parameter shall be set using the following masks:

Name	Mask value	Description
HTPT_TEMP_TYPE_CHAR_SUP	0x01	Indicate if the Temperature Type Char is supported
HTPT_INTERM_TEMP_CHAR_SUP	0x02	Indicate if Intermediate Temperature Char is supported
HTPT_MEAS_INTV_CHAR_SUP	0x04	Indicate if Measurement Interval Char is supported
HTPT_MEAS_INTV_IND_SUP	0x08	Indicate if the Measurement Interval Char supports sending of indications (will be taken in account only if the Measurement Interval Char is supported)
HTPT_MEAS_INTV_WR_SUP	0x10	Indicate if the Measurement Interval Char is writable (will be taken in account only if the Measurement Interval Char is supported)

Table 3.2-1 – Database creation feature flags

Example : If features = 0x1F, all optional features are supported.

Minimal and maximal measurement interval values will be used to initialize the value of the Valid Range descriptor. If these two values are not compliant (e.g. if both value are 0 or if minimal value is upper than maximal value), default values will be used. These default values can be modified in htpt.h.

2.4 API Messages

2.4.1 HTPT_ENABLE_REQ

Destination: TASK_HTPT

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection Index.
uint8_t	ntf_ind_cfg	Notification configuration (Bond Data to restore, see Table 2-2)

Response: HTPT_ENABLE_RSP

Description: This API message can be used after the connection with a peer device has been established in order to restore known device bond data.

The ntf_ind_cfg parameter is a bit field containing notification and indication configuration:

Name	Value	Description
HTPT_CFG_STABLE_MEAS_IND	0x01	Stable measurement interval indication enabled
HTPT_CFG_INTERM_MEAS_NTF	0x02	Intermediate measurement notification enabled
HTPT_CFG_MEAS_INTV_IND	0x04	Measurement interval indication

Table 2-2 - Notification and indication configuration (enum htpt_ntf_ind_cfg)

2.4.2 HTPT_ENABLE_RSP

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection Index.
uint8_t	status	Status code (see [5])

Response: None

Description: Inform application if restoring bond data for peer device succeed or not.

2.4.3 HTPT_TEMP_SEND_REQ

Destination: TASK_HTPT

Parameters:

Type	Parameters	Description
struct htp_temp_meas	temp_meas	Temperature Measurement value (see Table 4.3 – Temperature Measurement structure (struct htp_temp_meas))
uint8_t	flag_stable_meas	Indicate if the temperature measurement is stable: 0 (will be sent using the Temperature Measurement characteristic) or not: 1 (will be sent using the Intermediate Temperature characteristic)

Response: HTPT_TEMP_SEND_RSP

Description: This message is used by the application (which handles the temperature device driver and measurements) to send a temperature measurement through the Thermometer role. The *flag_stable_meas* determines if the temperature will be notified or indicated, and the rest of the parameters will determine what is present in the temperature structure in the PDU of the indication/notification.

Upon reception of this request, HTPT task will check if the necessary action (indication/notification) is possible on peer devices connected with the current configuration set by the Collectors in the HTS attributes.

If the action is possible, the temperature value is packed into a correct format in the appropriate attribute value, and the notification/indication request is sent to GATT to generate the required PDU for the peers.

2.4.4 HTPT_TEMP_SEND_RSP

Destination: TASK_APP

Parameters:

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

Response: None

Description: This message is used by HTPT to inform that Sent temperature measurement indication or notification is finished request.

2.4.5 HTPT_MEAS_INTV_UPD_REQ

Destination: TASK_HTPT

Parameters:

Type	Parameters	Description
uint16_t	meas_intv	Measurement Interval value

Response: HTPT_MEAS_INTV_UPD_RSP

Description: This message is used by the application to order the HTPT profile to generate an indication (if enabled) of the Measurement Interval Char on all connected peer device if indication configuration has been set.

This can be done as the application desires, at each connection, or if the measurement interval value has been modified locally (interface for this is not provided since a normal thermometer would have very few configurable UI elements and configuration should be done through Collector).

2.4.6 HTPT_MEAS_INTV_UPD_RSP

Destination: TASK_APP

Parameters:

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

Description: This message is used by HTPT to inform that measurement interval has been updated and peer devices have been informed.

2.4.7 HTPT_MEAS_INTV_CHG_REQ_IND

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection Index.
uint16_t	intv	New value of interval , written by Client

Response: HTPT_MEAS_INTV_CHG_CFM

Description: This message is used by the HTPT to inform the application that a peer device requests to update the measurement interval value.

If modification accepted, application uses the new value to either decide to stop periodic measurements if the value of the interval has changed from non 0 to 0, or the opposite, to start periodic measurements using the interval value, if the value has changed from 0 to non 0.

This message will only be issued if the new value that the Collector is trying to write is valid (within the Valid Range descriptor minimum and maximum values). If the value is not within range, this message is never received by the application because the HTPT will send an Error Response to the Collector with the 'Out of Range' code 0x80 and the new value will never be set.

2.4.8 HTPT_MEAS_INTV_CHG_CFM

Destination: TASK_HTPT

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection Index.
uint8_t	status	Status code (see [5])

Description: This message is used by application to confirm that measurement interval modification is accepted or not by the application. Write confirmation is sent back to peer devices that requests modification and if accepted, all other devices connected which have configured to receive measurement interval change indication will be informed about this modification.

2.4.9 HTPT_CFG_INDNTF_IND

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	conidx	Connection Index.
uint8_t	ntf_ind_cfg	Notification configuration (Bond Data to restore, see Table 2-2)

Description: This message is used to inform the Application that a peer device updated notification and indication configuration. This new configuration can be considered as bond data and can be stored into a non-volatile memory for future connection with this peer device.

3 Health Thermometer Profile Collector

This role is meant to be activated on the device that will collect the temperature measurements from the Thermometer. It implies it is a GAP Central. The FW task for this role will discover the HTS present on the peer Server, after establishing connection, and will allow configuration of the HTS attributes if so required. Please refer to "htpc_task.h" for implementation of this API.

This task has 3 possible states: FREE, IDLE, BUSY.

Important Note: The TASK_HTPC 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 HTPC task instance.

The term TASK_HTPC_IDX will be used in the rest of the document to refer to any instance of the Health Thermometer Profile Collector Role Task. The term TASK_HTPC will refer to the first instance of this task.

3.1 Initialization

During the initialization phase of the device, to use the Health Thermometer Client task, the HTPC task has to be allocated using GAPM API. Application has to send GAPM_PROFILE_TASK_ADD_CMD [6].

3.2 HTPC_ENABLE_REQ

Destination: TASK_HTPC_IDX

Parameters:

Type	Parameters	Description
uint8_t	con_type	Connection type: 1st discovery(configuration) or normal connection.
struct hts_content	hts	HTS details (see Table 4.4-3)

Response: HTPC_ENABLE_RSP

Description: This API message is used for enabling the Collector role of the Health Thermometer 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 HTS details on peer.

The connection type may be 0 = Connection for discovery/initial configuration or 1 = Normal connection. This difference has been made and Application would handle it in order to not discover the HTS on the Thermometer at every connection, but do it only once and keep the discovered details in the Collector device between connections. Configuration can be done during a normal connection also, but since most use cases allow Thermometer to disconnect the link once all measurements have been sent to Collector, the Collector may not have the time for it.

If it is a discovery /configuration type of connection, the *hts* and *dis* parameters are useless, they will be filled with 0's. Otherwise they will contain pertinent data which will be kept in the Collector 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 HTS and DIS content in the environment and registering HTPC in GATT to receive the indications and notifications for the known attribute handles in HTS that would be notified/indicated. For a discovery connection, discovery of the peer HTS is started and the response will be sent at the end of the discovery with the discovered attribute details.

The Task for this profile role will go from IDLE state to CONNECTED state for a normal connection, and to DISCOVERING state for a discovery/configuration type of connection.

3.3 HTPC_ENABLE_RSP

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint8_t	status	Enable status: discovery error code if anything goes wrong during a configuration type connection. (see [5])
struct hts_content	hts	HTS discovered details (see Table 4.4-3)

Response: None

Description: This API message is used by the Collector to either send the discovery results of HTS on the Thermometer and confirm enabling of the Collector role, or to simply confirm enabling of Collector role if it is a normal connection and the attribute details are already known.

3.4 HTPC_RD_CHAR_REQ

Destination: TASK_HTPC_IDX

Parameters:

Type	Parameters	Description
uint8_t	char_code	Code for which characteristic to read. (see Table 4.6 – Read Request Codes)

Response: HTPC_RD_CHAR_RSP

Description: This API message is used by the application to send a GATT_READ_CHAR_REQ with the parameters deduced from the *char_code*. Upon reception of this message, HTPC checks whether the parameters are correct, then if the handle for the characteristic is valid (not 0x0000) and the request is sent to GATT. When the peer has responded to GATT, and the response is routed to HTPC, the HTPC_RD_CHAR_RSP message will be generically built and the Application must be able to interpret it based on the read request it made. And error status is also possible either for the Read procedure or for the application request, in the second case, the HTPC_ERROR_IND message is sent to Application.

No parsing intelligence of the received response is added in this API handler, so all the work of interpretation must be added in the Application depending of its request and use of the response.

3.5 HTPC_RD_CHAR_RSP

Destination: TASK_APP

Parameters:

Type	Parameters	Description
struct att_info_data	data	Structure containing the read value • uint16_t: attribute handle • uint16_t: data length • uint8_t: status code (see [5]) • uint8_t[]: data

Response: None

Description: This API message is used by the Collector role to inform the Application of a received read response. The status and the data from the read response are passed directly to Application, which must interpret them based on the request it made.

3.6 HTPC_HEALTH_TEMP_NTF_CFG_REQ

Destination: TASK_HTPC_IDX

Parameters:

Type	Parameters	Description
uint16_t	cfg_val	Configuration value.
uint8_t	char_code	Code for which characteristic to configure in ntf/ind

Response: HTPC_HEALTH_TEMP_NTF_CFG_RSP

Description: This API message is used by the application to send a GATT_WRITE_CHAR_REQ with the parameters deduced from the *char_code* and *cfg_val.* The definitions for the different codes for characteristics that can be configured to indicate/notify are in htpc.h. Upon reception of this message, HTPC checks whether the parameters are correct, then if the handle for the characteristic is valid (not 0x0000) and the request is sent to GATT. When the peer has responded to GATT, and the response is routed to HTPC, the HTPC_WR_CHAR_RSP message will be generically built and sent to Application. An error status is also possible either for the Write procedure or for the application request, in the second case, the HTPC_ERROR_IND message is sent to Application.

3.7 HTPC_HEALTH_TEMP_NTF_CFG_RSP

Destination: TASK_APP

Parameters:

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

Description: This API message is used by the Collector role to inform the Application of a received write response. The status and the data from the write response are passed directly to Application, which must interpret them based on the request it made.

3.8 HTPC_WR_MEAS_INTV_REQ

Destination: TASK_HTPC_IDX

Parameters:

Type	Parameters	Description
uint16_t	intv	Measurement interval value to write

Response: HTPC_WR_MEAS_INTV_RSP

Description: This API message is used by the application to send a GATT_WRITE_CHAR_REQ to the HTS Measurement Interval Char. In the Thermometer, with the new interval value in *intv.* Upon reception of this message, HTPC checks whether the parameters are correct, then if the handle for the characteristic is valid (not 0x0000) and whether it is writable or not - if all OK, the request is sent to GATT, otherwise a HTPC_ERROR_IND message is built for the Application. When the peer has responded to GATT, and the response is routed to HTPC, the HTPC_WR_CHAR_RSP message will be generically built and sent to Application. An error status is also possible for the Write procedure, it will be sent through that same message. It is the application's responsibility to write a measurement interval value that respects the valid range in HTS characteristics.

3.9 HTPC_WR_MEAS_INTV_RSP

Destination: TASK_APP

Parameters:

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

Description: This API message is used by the Collector role to inform the Application of a received write response. The status and the data from the write response are passed directly to Application, which must interpret them based on the request it made.

3.10 HTPC_TEMP_IND

Destination: TASK_APP

Parameters:

Type	Parameters	Description
struct htp_temp_meas	temp_meas	Received temperature measurement value
bool	stable_meas	Flag indicating if it is a stable(1) or intermediary (0)measurement.

Description: This API message is used by the Collector role to inform the Application of a received temperature value, either by notification *(*stable_meas *= false* → intermediate*)* or indication *(*stable_meas *= true* → stable*).* No confirmation of reception is needed because the GATT sends it directly to the peer.

3.11 HTPC_MEAS_INTV_IND

Destination: TASK_APP

Parameters:

Type	Parameters	Description
uint16_t	intv	Interval value that is being indicated.

Response: None

Description: This API message is used by the Collector role to inform the Application of a received Measurement Interval Char. Indication and the value it indicates. This value should be used by the Application as seen fit. No response is necessary (the GATT sends the necessary confirmation to the Indication PDU).

4 Miscellaneous

Type	Parameters	Description
uint32_t	temp	Temperature value
struct prf_date_time	time_stamp	Measurement Time
uint8_t	flags	Features flag
uint8_t	type	Temperature Type

Table 4.4-1 – Temperature Measurement structure (struct htp_temp_meas)

Type	Parameters	Description
uint16_t	year	Year
uint8_t	month	Month
uint8_t	day	Day
uint8_t	hour	Hour
uint8_t	min	Minutes
uint8_t	sec	Seconds

Table 4.4-2 – Time Stamp structure (struct prf_date_time)

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	chars[HTPC_CHAR_HTS_MAX] Indexes: HTPC_CHAR_HTS_TEMP_MEAS HTPC_CHAR_HTS_TEMP_TYPE HTPC_CHAR_HTS_INTM_TEMP HTPC_CHAR_HTS_MEAS_INTV	Structure containing handles and properties of characteristics /// characteristic handle uint16_t char_hdl; /// value handle uint16_t val_hdl; /// characteristic properties uint8_t prop;
struct prf_char_desc_inf	desc[HTPC_DESC_HTS_MAX] Indexes: HTPC_DESC_HTS_TEMP_MEAS_CLI_CFG HTPC_DESC_HTS_INTM_MEAS_CLI_CFG HTPC_DESC_HTS_MEAS_INTV_CLI_CFG HTPC_DESC_HTS_MEAS_INTV_VAL_RGE	Structure containing descriptor handle /// Descriptor handle uint16_t desc_hdl;

Table 4.4-3 – HTS Content (struct htpc_hts_content)

Name	Description
HTPC_RD_HTS_TEMP_TYPE	Read Temperature Type value
HTPC_RD_HTS_MEAS_INTV	Read Measurement Interval value
HTPC_RD_HTS_TEMP_MEAS_CLI_CFG	Read Temperature Measurement Client Cfg. Desc value
HTPC_RD_HTS_INTM_TEMP_CLI_CFG	Read Intermediate Temperature Client Cfg. Desc value
HTPC_RD_HTS_MEAS_INTV_CLI_CFG	Read Measurement Interval Client Cfg. Desc value
HTPC_RD_HTS_MEAS_INTV_VAL_RGE	Read Measurement Interval Valid Range value

Table 4.4-4 – Read Request Codes

5 Abbreviations

Abbreviation	Original Terminology
API	Application Programming Interface
BLE	Bluetooth Low Energy
DIS	Device Information Service
HTPC	Health Thermometer Profile Collector
HTPT	Health Thermometer Profile Thermometer
HTS	Health Thermometer Service
GAP	Generic Access Profile
GATT	Generic Attribute Profile
RW	RivieraWaves

References

[1]	**Title**	Health Thermometer Profile		
	Reference	HTP_SPEC_V10		
	Version	V10r00	**Date**	May 24th 2011
	Source	Bluetooth SIG – Medical Working Group		

[2]	**Title**	Health Thermometer Service		
	Reference	HTS_SPEC_V10		
	Version	V10r00	**Date**	May 24th 2011
	Source	Bluetooth SIG – Medical Working Group		

[3]	**Title**	Health Thermometer Profile (HTP) 1.0		
	Reference	HTP.TS.1.0.0		
	Version	1.0.0	**Date**	May 24th 2011
	Source	Bluetooth SIG		

[4]	**Title**	Health Thermometer Service (HTS) 1.0		
	Reference	HTS.TS.1.0.0		
	Version	1.0.0	**Date**	May 24th 2011
	Source	Bluetooth SIG		

[5]	**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		

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