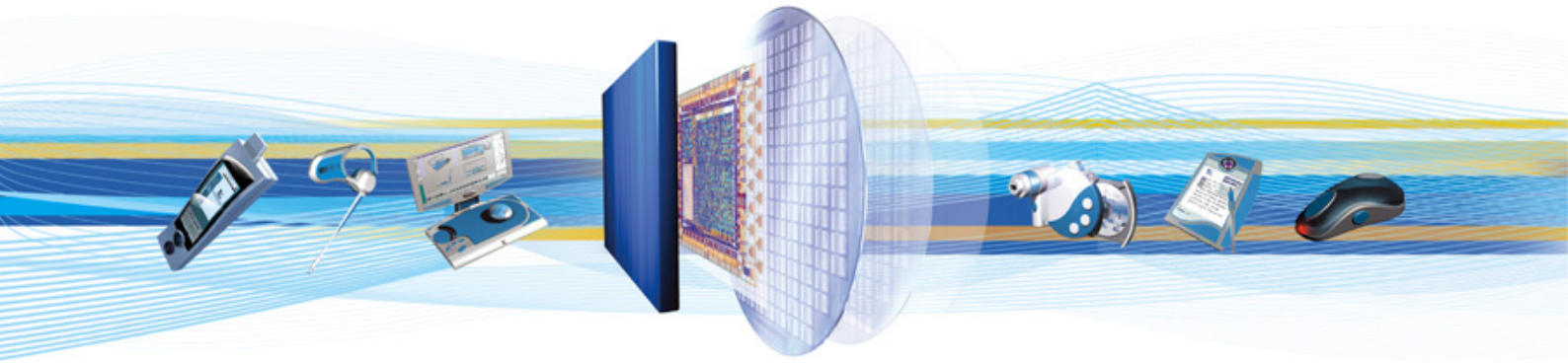
CSR Synergy Bluetooth 18.2.2

BT LE Thermometer Server Service

API Description

January 2012



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

Contents

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

List of Figures

List of Tables

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

1 Introduction

1.1 Introduction and Scope

This document describes the functionality and message interface provided by CSR Synergy Bluetooth for using the Bluetooth Low Energy(BLE) Health Thermometer service server – as specified by the Bluetooth® Special Interest Group (SIG).

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

2 Description

2.1 Introduction

The BLE Health Thermometer Server Service extends the GATT task with the functionality needed to make a simple Health Thermometer server device. The server is constructed to give the implementer the easiest possible way to implement the health thermometer service without having to care about the full GATT API. This means that after activation, the server task will run in the background and keep serving incoming connection on LE until it is deactivated.

The Health thermometer server makes it possible to implement the health thermometer server profile easily and provides a simple interface for the application to subscribe to relevant events. It handles both the setup of the database and the connection handling. The following Low Energy services are supported by the database in the server:

- Thermometer service
- Battery Service

The functionality of each of the services are defined in the Bluetooth Low Energy Thermimity profile documentation as provided by the Bluetooth SIG.

2.2 Requirements

In order for the health thermometer Server to work, the Synergy Bluetooth stack needs to be compiled with CSR_BT_LE_ENABLE=1 . Besides this, the GATT task needs to be included and enabled in the scheduler.

The Bluetooth chip needs to be compatible with the Synergy Bluetooth Low Energy enabled host stack.

2.3 Reference Model

The reference model for a Bluetooth LE Health Thermometer server is depicted in Figure 1 below. The server is placed between the GATT API and the application in order to provide a simplified API for the application. The Application does not have do have any communication directly with the GATT task if only the health thermometer server service is needed.

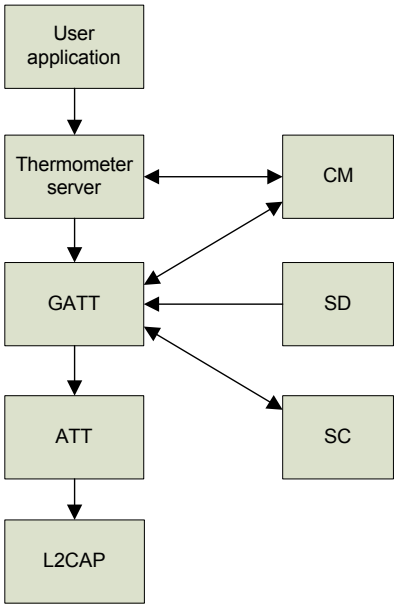


Figure 1: Health Thermometer Server reference model

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

3 Interface Description

This chapter documents the public API of the LE health thermometer Server service. The implementation follows the normal guidelines for Synergy Bluetooth task implementations.

3.1 Activate

In order to use the health thermometer server, it needs to be activated. Activating the server will make it start advertising on the LE radio and wait for incoming connections.

If a device connects to the server, then the advertisement will stop. When the device disconnects, the health thermometer server will start advertising on the LE radios again.

In order to provide the option to store Characteristic Client Configurations in a persistent store, the deactivate confirm signal returns a client configuration datablob. When you re-activate the server, it is possible to provide a pointer to this datablob and thereby restore the previous Characteristics Client Configuration state. The application does not need to understand the client config data in the datapointer.

It is possible to provide an event mask to tell the server which type of event notifications the app would like to receive. See the csr_therm_srv_prim.h for definition of the event mask types

To send the `CSR_BT_THERM_SRV_ACTIVATE_REQ` primitive, the `CsrBtThermSrvActivateReqSend()` function is used. The function takes the arguments described in Table 1.

Type	Argument	Description
CsrSchedQid	appHandle	Application handle for the app initializing the thermometer server
CsrUint16	clientConfigSize	If a client config data blob is added, this is the size in bytes
CsrUint8	*clientConfig	Client config data blob returned in a previous deactivation
CsrBtThermSrvEventMask	eventMask	Mask indicating which events the app would like to subscribe to

Table 1: Arguments for `CsrBtThermSrvAvtivateReqSend` function

When the registration request has been processed, a `CSR_BT_THERM_SRV_ACTIVATE_CFM` primitive will be sent back to the application. The primitive members are described in Table 2.

Type	Member	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_ACTIVATE_CFM
CsrBtGattId	gattId	The ID the GATT profile uses to identify the thermometer server
CsrUint16	dbStartHandle	The database start handle allocated for the thermometer server
CsrUint16	dbEndHandle	The database end handle allocated for the thermometer server
CsrBtResultCode	resultCode	Result code returned by the supplier in resultSupplier
CsrBtSupplier	resultSupplier	The result supplier – CSR_BT_SUPPLIER_THERM_SRV is success

Table 2: Members in a CSR_BT_THERM_SRV_ACTIVATE_CFM primitive

3.2 Deactivate

The server can be deactivated by the application at any time during use. When deactivated, the server will disconnect any existing connections and clean up state information.

In order to provide the option to store Characteristic Client Configurations in a persistent store, the deactivate confirm signal returns a client config datablob. When you re-activate the server, it is possible to provide a pointer to this datablob and thereby restore the previous Characteristics Client Configuration state.

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

To send the CSR_BT_THERM_SRV_DEACTIVATE_REQ primitive, the CsrBtThermSrvDeactivateReqSend() function is used. The function takes the arguments described in **Error! Reference source not found.Error! Reference source not found.**.

When the deactivation request has been processed, a CSR_BT_THERM_SRV_DEACTIVATE_CFM primitive will be sent back to the application. The primitive members are described in Table 3.

Type	Member	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_DEACTIVATE_CFM
CsrUint16	clientConfigSize	Size of the client config data blob returned
CsrUint8	*clientConfig	Client config data blob for later re-addition in another activate
CsrBtResultCode	resultCode	Result code returned by the supplier in resultSupplier
CsrBtSupplier	resultSupplier	The result supplier – CSR_BT_SUPPLIER_THERM_SRV is success

Table 3: Members in a CSR_BT_THERM_SRV_DEACTIVE_CFM primitive

3.3 Connect and Disconnect

If a peer device connects to the server, a connect indication signal is sent to the application. This provides information about the connection, the peer device addres and type. No reply is needed for this signal.

The parameters for the CSR_BT_THERM_SRV_CONNECT_IND primitive are described in Table 4.

Type	Argument	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_CONNECT_IND
CsrBtTypedAddr	deviceAddr	
CsrBtConnId	btConnId	
CsrBtResultCode	resultCode	Result code returned by the supplier in resultSupplier
CsrBtSupplier	resultSupplier	The result supplier – CSR_BT_SUPPLIER_THERM_SRV is success

Table 4: Members in a CSR_BT_THERM_SRV_CONNECT_IND primitive

If a peer device disconnects from the server, a disconnect indication signal is sent to the application. This provides information about the reason for the disconnect and who disconnected. No reply is needed for this signal.

The parameters for the CSR_BT_THERM_SRV_DISCONNECT_IND primitive are described in Table 5.

Type	Argument	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_DISCONNECT_IND
CsrBtConnId	btConnId	
CsrBtTypedAddr	deviceAddr	
CsrBtResultCode	reasonCode	Reason code returned by the supplier in reasonSupplier
CsrBtSupplier	reasonSupplier	The reason supplier – CSR_BT_SUPPLIER_THERM_SRV is success

Table 5: Members in a CSR_BT_THERM_SRV_DISCONNECT_IND primitive

3.4 Update Temperature

In order for the server to have the latest information about the Temperature level, the application needs to send down this information whenever it thinks it needs updating. The interval for updating this is not set to anything specific as it is a task of the application to decide on this matter. If no temperature value is sent to the thermometer service server task, the default value of 0 is provided to peer devices requesting this value.

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

To send the CSR_BT_THERM_SRV_UPDATE_TEMPERATURE_REQ primitive, the CsrBtThermSrvUpdateTemperatureReqSend() function is used. The function takes the arguments described in Table 6.

Type	Argument	Description
CsrUint16	tempDataSize	The size of the temperature Data blob in bytes
CsrUint8*	tempData	The temperature Data pointer. Byte 0 is the temperature type (Celsius 0x00 Fahrenheit: 0x01) Rest of the data is the temperature as float packed as byte-data. The format of the value shall follow the IEE-11073 format to be compliant with the LE thermometer specification.

Table 6: Arguments for `CsrBtThermSrvUpdateTemperatureReqSend` function

When the update temperature request has been processed, a CSR_BT_THERM_SRV_UPDATE_TEMPERATURE_CFM primitive will be sent back to the application. The primitive members are described in Table 7.

Type	Member	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_UPDATE_TEMPERATURE_CFM
CsrBtResultCode	resultCode	Result code returned by the supplier in resultSupplier
CsrBtSupplier	resultSupplier	The result supplier – CSR_BT_SUPPLIER_THERM_SRV is success

Table 7: Members in a CSR_BT_THERM_SRV_UPDATE_TEMPERATURE_CFM primitive

3.5 Update Battery Level

In order for the server to have the latest information about the battery level and state, the application needs to send down this information whenever it thinks it needs updating. The interval for updating this is not set to anything specific as it is a task of the application to decide on this matter.

If no battery value is sent to the thermometer server task, the default value of 0 is provided to peer devices requesting this value.

To send the CSR_BT_THERM_SRV_UPDATE_BATT_LEVEL_REQ primitive, the CsrBtThermSrvUpdateBattLevelReqSend() function is used. The function takes the arguments described in Table 8.

Type	Argument	Description
CsrUint16	battLevel	The new battery level value
CsrBtThermSrvBatteryMask	battMask	Bitmask identifying the state of the battery. Bit 0-1: battery present Bit 2-3: battery discharging Bit 4-5: battery charging Bit 6-7: battery state critical Possible values are defined in csr_bt_therm_srv_prim.h
CsrUint8	serviceRequired	Service required field values as defined in the file csr_bt_therm_srv_prim.h.

Table 8: Arguments for `CsrBtThermSrvUpdateBattLevelReqSend` function

When the Update Battery Level request has been processed, a CSR_BT_THERM_SRV_UPDATE_BATT_LEVEL_CFM primitive will be sent back to the application. The primitive members are described in Table 9.

Type	Member	Description

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

Type	Member	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_UPDATE_BATT_LEVEL_CFM
CsrBtResultCode	resultCode	Result code returned by the supplier in resultSupplier
CsrBtSupplier	resultSupplier	The result supplier – CSR_BT_SUPPLIER_THERM_SRV is success

Table 9: Members in a CSR_BT_THERM_SRV_UPDATE_BATT_LEVEL_CFM primitive

3.6 Write Event

When the application is activated, it is possible to specify a bitmask telling which events it want to subscribe to. Whenever one of the events are triggered in the server, then the it will send off the subscribed event to the application.

The write event is triggered when the peer device tries to write to one of the writable handles in the database where IRQ is enabled.

The parameters for the `CSR_BT_THERM_SRV_WRITE_EVENT_IND` primitive are described in Table 10.

Type	Argument	Description
CsrBtThermSrvPrim	type	Signal identity – always set to CSR_BT_THERM_SRV_WRITE_EVENT_IND
CsrUint16	valueHandle	Database handle where the value is written to
CsrUint16	valueSize	Size in bytes of the value written to the database
CsrUint8	*value	The actual data pointer with the value

Table 10: Members in a CSR_BT_THERM_SRV_WRITE_EVENT_IND primitive

.

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
BLE	Bluetooth Low Energy
LE	Common name for the Low Energy Bluetooth® radio.
BR/EDR	Basic Rate / Enhanced Data Rate. Common name for the standard Bluetooth® radio technology.
CSR	Cambridge Silicon Radio
DUN	Dial Up Networking
FCS	Frame Check Sequence, 16bit CRC used in L2CAP
FTC	File Transfer Client
FTP	File Transfer Protocol
FTS	File Transfer Server
GOEP	Generic Object Exchange Protocol
HCI	Host Controller Interface
L2CAP	Logical Link Control and Adaption Protocol
MSC	Message Sequence Chart
OBEX	Object Exchange Protocol
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0
2	23 JAN 12	Ready for release 18.2.2

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information

© Cambridge Silicon Radio Limited 2012
This material is subject to CSR's non-disclosure agreement.