Software User Manual

BROADCOM®

BSA Sample Apps

BSA on iMX6 Development Board

BSA-SERVER-SWUM101-R

Irvine, CA 92617

March 08, 2016

Revision History

Revision	*Date*	*Change Description*
BSA-SERVER-SWUM101-R	03/08/16	**Updated:** • Figure 1: “BSA Usage Structure,” on page 6 • “Setup Target” on page 6 • “Running BSA Server and Client” on page 7 **Added:** • “Running BSA Client to Stream (A2DP)” on page 9
BSA-SERVER-SWUM100-R	10/23/15	Initial release

Table of Contents

About This Document

Purpose and Audience

This document provides instructions to run BSA server and sample client application on iMX6 development board. BSA server requires Broadcom Corporation Bluetooth device to interface.

This document is for software developers who are using BSA server with Broadcom Bluetooth device to test BSA sample client applications. With the sample BSA client applications, users could evaluate the BSA APIs and verify the hardware interfaces.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	*Description*
Bold	User input and actions: for example, type **exit**, click **OK**, press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

Technical Support

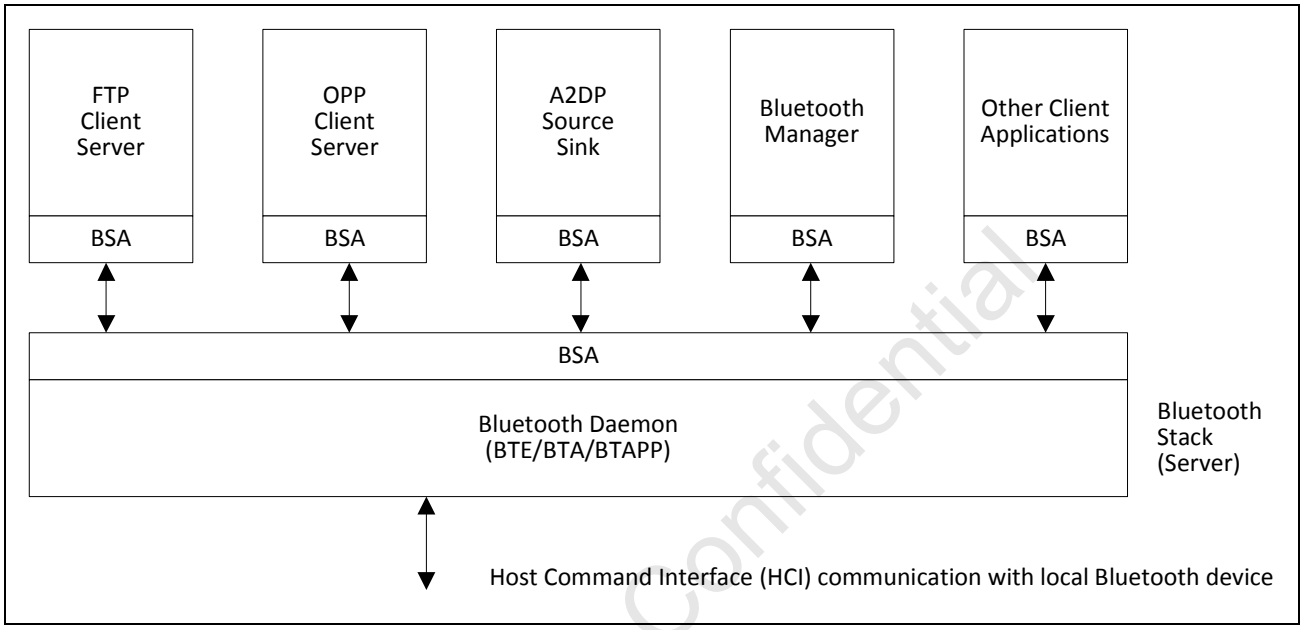
Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Introduction

One or more client applications can connect to the server to use Bluetooth services/profiles (FTP, AG, A2DP, and so forth.) The BSA usage structure is shown in Figure .

Figure 1: BSA Usage Structure



The communication between clients and the servers is based on sockets (such as local, named, and Linux® sockets) and on named first in and first out (FIFO) pipes.

Setup Target

Create a bootable SD Card with released Linux software as described in “Wi-Fi/BT EVK for i.MX6 Quick Start Guide (Linux)”, https://community.broadcom.com/community/linux. The Quick Start Guide “Preparing Bootable SD Card for i.MX6 with Wi-Fi/BT EVK” describes steps to create an SD Card, follow this section to create bootable SD Card. Steps in section Quick Start Guide “Copying/Replacing Files on SD Card” describes how to update image and setup to run WiFi and Bluetooth.

Configure Hardware to Run BSA

If the setup had additional SD card extender then step 7 in section 2.1 of the Quick Start Guide needs to be followed.

```
[7]Power on platform and interrupt at u-boot. Type in following commands to set correct DTB file:
    => setenv fdt_file imx6sx-sdb-murata-v2.dtb
    => saveenv
    => reset (causes platform to boot kernel)
```

Use the below commands to set the reset for the Bluetooth, BT_REG_ON is GPIO(6,11) = GPIO 171.

```
echo 171 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio171/direction
echo 0 > /sys/class/gpio/gpio171/value
sleep 1
echo 1 > /sys/class/gpio/gpio171/value
```

Running BSA Server and Client

To run the BSA server and client, do the following:

1. Copy bsa_server and app_manager to desired directory (e.g.: ~/).
2. Copy patch file to same directory or locate it in the file system.
3. Open 2 terminals to run BSA server and client applications.

Running BSA Server

On one of the terminals, run the bsa_server: `bsa_server –d /dev/tty… -p <patch_file> -all <debug level>`

For Type_1DX (BCM4343W) or Type 1BW (BCM43340) module mounted on iMX6 Ultralite or iMX6SX_SABRE with SDIO adapter use the following command:

For Type_1DX:

```
./bsa_server -d /dev/ttymxc2 -p BCM43430A1.Type_1DX.hcd -all 2
```

For Type_1BW:

```
./bsa_server -d /dev/ttymxc2 -p BCM43341B0.1BW.hcd -all 2
```

Running BSA Client to Connect

After starting the BSA Server on one of the terminals, on the second terminal run the app_manager (client application)

```
./app_manager
```

The following menu options will be displayed:

```
Bluetooth Application Manager Main menu:
    1 => Abort Discovery
    2 => Discovery
    3 => Discovery test
    4 => Bonding
    5 => Cancel Bonding
    6 => Remove device from security database
    7 => Services Discovery (all services)
    8 => Device Id Discovery
    9 => Stop Bluetooth
    10 => Restart Bluetooth
    11 => Accept Simple Pairing
    12 => Refuse Simple Pairing
    13 => Act As HID Keyboard (SP passkey entry)
    14 => Read Device configuration
```

```
    15 => Read Local Out Of Band data
    16 => Enter remote Out Of Band data
    17 => Set device visibility
    18 => Set device BLE visibility
    19 => Set AFH Configuration
    20 => Set Tx Power Class2 (specific FW needed)
    21 => Set Tx Power Class1.5 (specific FW needed)
    22 => Change Dual Stack Mode (currently:DUAL_STACK_MODE_BSA)
    23 => Set Link Policy
    24 => Enter Passkey
    25 => Get Remote Device Name
    96 => Kill BSA server
    97 => Connect to BSA server
    98 => Disconnect from BSA server
    99 => Quit
Select action =>
```

Use option 2 to discover BT devices.

```
Select action => New Discovered device:0
    Bdaddr:64:a3:cb:5b:69:f0
    Name:Keloglan
    ClassOfDevice:7a:02:0c => Phone
    Services:0x00000000 ()
    Rssi:-55
BSA_trace 786@ 01/01 00h:55m:05s:725ms: COMPLETE_LOCAL_NAME_TYPE:
BSA_trace 787@ 01/01 00h:55m:05s:725ms:    0000: 4b 65 6c 6f 67 6c 61 6e
Keloglan
BSA_trace 788@ 01/01 00h:55m:05s:725ms: COMPLETE_16BITS_UUID_TYPE:
BSA_trace 789@ 01/01 00h:55m:05s:725ms:    0000: 00 12 1f 11 2f 11 0a 11 0c 11 32 11
..../.....2.
BSA_trace 790@ 01/01 00h:55m:05s:725ms: COMPLETE_32BITS_UUID_TYPE:
BSA_trace 791@ 01/01 00h:55m:05s:726ms:    0000:
BSA_trace 792@ 01/01 00h:55m:05s:726ms: COMPLETE_128BITS_UUID_TYPE:
BSA_trace 793@ 01/01 00h:55m:05s:726ms:    0000: fe ca ca de af de ca de de fa ca de 00 00 00
00   ................
BSA_trace 794@ 01/01 00h:55m:05s:726ms: MANUFACTURER_SPECIFIC_TYPE:
BSA_trace 795@ 01/01 00h:55m:05s:726ms:    0000: 00 4c 02 24 02 00 00 00 00 00 00 00 00 00 00
00   .L.$............
BSA_trace 796@ 01/01 00h:55m:05s:726ms:    0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00   ................
BSA_trace 797@ 01/01 00h:55m:05s:726ms:   0020: 00 00 00 00 00 00                    ......
    inq_result_type:1 ble_addr_type:0 device_type:1
    ……..
```

Use option 4 to bond/pair with a device then select a device to bond from the menu.

```
Select action => BSA_trace 880@ 01/01 01h:01m:28s:532ms: bsa_sec_event_hdlr event:0
DEBUG: app_mgr_security_callback: event:0
DEBUG: app_mgr_security_callback: BSA_SEC_LINK_UP_EVT bd_addr: 64:a3:cb:5b:69:f0
DEBUG: app_mgr_security_callback: ClassOfDevice:7a:02:0c => Phone
DEBUG: app_mgr_security_callback: LinkType: 1
BSA_trace 881@ 01/01 01h:01m:29s:561ms: bsa_sec_event_hdlr event:6
DEBUG: app_mgr_security_callback: event:6
DEBUG: app_mgr_security_callback: BSA_SEC_SP_CFM_REQ_EVT
DEBUG: app_mgr_security_callback:     Remote device:Keloglan
DEBUG: app_mgr_security_callback:     bd_addr: 64:a3:cb:5b:69:f0
DEBUG: app_mgr_security_callback:     ClassOfDevice:7a:02:0c => Phone
DEBUG: app_mgr_security_callback:     Just Work:FALSE
```

```
DEBUG: app_mgr_security_callback:     Numeric Value:226243
DEBUG: app_mgr_security_callback:     You must accept or refuse using menu
```

Pairing must be accepted by the paired device.

From the app_manager select option 11 => Accept Simple Pairing

```
11
DEBUG: app_mgr_sp_cfm_reply:
BSA_trace 882@ 01/01 01h:01m:35s:942ms: BSA_SecSpCfmReplyInit
BSA_trace 883@ 01/01 01h:01m:35s:942ms: BSA_SecSpCfmReply
Bluetooth Application Manager Main menu:
…..
Select action => BSA_trace 884@ 01/01 01h:01m:36s:121ms: bsa_sec_event_hdlr event:3
DEBUG: app_mgr_security_callback: event:3
DEBUG: app_mgr_security_callback: BSA_SEC_AUTH_CMPL_EVT (name=Keloglan,success=1)
DEBUG: app_mgr_security_callback:     bd_addr:64:a3:cb:5b:69:f0
DEBUG: app_mgr_security_callback:     LinkKey:fe:f1:70:c1:47:27:0a:a3:8b:50:42:67:b7:30:7f:05
Update name with Keloglan
Update link key
Update class-of-device [0x7A-0x02-0x0C]
DEBUG: app_mgr_write_remote_devices: app_xml_write_db ok
BSA_trace 885@ 01/01 01h:01m:39s:106ms: bsa_sec_event_hdlr event:1
DEBUG: app_mgr_security_callback: event:1
DEBUG: app_mgr_security_callback: BSA_SEC_LINK_DOWN_EVT bd_addr: 64:a3:cb:5b:69:f0
DEBUG: app_mgr_security_callback: Reason: 22
DEBUG: app_mgr_security_callback: LinkType: 1
```

The bonding/pairing only connects to the device then disconnects.

Running BSA Client to Stream (A2DP)

After starting the BSA Server on one of the terminals and the on the app_manager on the second terminal, on the third terminal run the app_av (client application)

```
./app_av
```

The app_av client application will be used to scan the BT speaker to stream sound then connect to the BT speaker. After successful connection, app_av client application will be used to stream the sound.

By default the bsa_server debug level is set to 5 (highest level between 0 and 5.) In order to suppress the log messages, perhaps debug level of 1 (-all 1) could be used.

```
./bsa_server -d /dev/ttymxc2 -p BCM43341B0.1BW.hcd -all 1
```

The app_manager client application is used to accept the simple connection between BT speaker and BSA server.

```
./app_manager
```

Run the app_av BSA client application to setup the BT speaker connection and the stream audio.

```
./app_av
```

Below are the steps to connect the BT speaker.

Using app_av:

```
1.  2 => Start Discovery
2.  6 => AV Open (Connect)
    2.1 1 Device found in last discovery
    (If it was connected previously, “0 Device from XML database (already paired)” could be used.)
    Dev:4
        Bdaddr:04:88:e2:89:62:16
        Name:BeatsPill
        ClassOfDevice:24:04:14 => Audio/Video
        Rssi:-47
        VidSrc:1 Vid:0x00CC Pid:0x2000 Version:0x0100
    Select device => 4 (Select the sink device to stream)
```

Using app_manager BSA client application accept the Accept Simple Pairing:

```
3.  11 => Accept Simple Pairing
```

Using app_av BSA client application

```
4.  4 => AV Register (Create local source point)
```

The "9 => AV Play Tone" menu option could be used to play tone using app_av.

The "14 => AV Stop" menu option could be used to stop the tone (streaming) using app_av.

To steam WAV files using app_av BSA client application, place the WAV files in "./test_files/av/" relative to where app_av is located.

The "11 => AV Play File" menu option could be used to play tone using app_av.

```
Play list:
    0 : ./test_files/av/air_raid.wav
        codec(PCM) ch(1) bits(8) rate(11025)
    1 : ./test_files/av/greatfuldeadSugaree.wav
        codec(PCM) ch(2) bits(16) rate(11025)
    2 : ./test_files/av/greatfuldeadSugareeHiQ.wav
        codec(PCM) ch(2) bits(16) rate(44100)
    3 : ./test_files/av/greatfuldeadSugareeLowQ.wav
        codec(PCM) ch(2) bits(16) rate(11025)
    4 : ./test_files/av/greatfuldeadSugareeMidQ.wav
        codec(PCM) ch(2) bits(16) rate(22050)
```

Select a WAV file to stream from the provided list of WAV files.

BSA Client Applications

The following BSA regular applications are summarized in the following table. They would be executed as described above for the app_manager BSA client application.

Name	*Description*
app_3d	This application is used to connect 3D stereo glasses
app_ag	Audio Gateway (AG) application. AG applications (usually located in cellular phones) are used to connect to mono headsets.
app_av	This application is used to stream music from an AV Source (AVS) profile.
app_avk	This application is used to receive a music steam from an AV sink (AVK) profile.
app_ble	This application shows how to use BLE client/server module BSAs.
app_ble_blp	This application is sample code for the BLE blood-pressure collector.
app_ble_cscc	This application is sample code for the BLE cycling speed and cadence collector.
app_ble_hrc	This application is sample code for the BLE heart rate controller.
app_ble_htp	This application is sample code for the BLE health thermometer collector.
app_ble_pm	This application is sample code for the BLE proximity monitor.
app_ble_rscc	This application is sample code for the BLE running speed and cadence collector.
app_ble_wifi	This sample application makes it easier to set up a Wi-Fi Direct connection between peers by exchanging Wi-Fi Direct information over BLE.
app_dg	Data Gateway (DG). Two Bluetooth devices can connect using this application to exchange data via serial communications port emulation.
app_fm	This application shows how to use the FM radio APIs.
app_ftc	FTP client (FTC). FTP clients can connect and access (browse/read/write) files located on the FTP server.
app_fts	File Transfer Server (FTS). The FTS can connect and access (browse/read/write) files located on the FTP server.
app_hd	Demonstrates Bluetooth HID device and remote control functionality.
app_headless	Shows how the Headless mode is controlled.
app_hd	Connects Human Interface Devices (HIDs) such as a keyboard, mouse, or remote control.
app_hh	Connects Human Interface Devices (HIDs) such as a keyboards, mice, or remote controls.
app_hl	This application is used to connect the health devices.
app_hs	Headset (HS) application. HS applications are used to connect to cellular phones which run AG applications.
app_manager	The main regular application. Governs security and device management. It must always be running.
app_mce	This application provides the Bluetooth MAP client (Message Client Equipment [MCE]) function.
app_opc	OP client (OPC). OP clients can exchange (send/receive) files with the OPS.
app_ops	Object Push (OP) server (OPS). OP clients can exchange (send/receive) files with the OP server.
app_pan	This application provides the test framework for Bluetooth PAN functionality.
app_pbc	This application provides the Bluetooth PBAP client (PBC) function.

Name	*Description*
app_pbs	Phone Book (PB) server (PBS). PB clients can connect to the PBS to download/upload phone books.
app_sac	This application provides the Bluetooth SIM Access Profile (SAP) client function.
app_sc	This application provides the Bluetooth SAP client (SC) function.
app_switch	This application demonstrates the sequence of operations necessary to implement role switching between AV/AG and AVK/HF.
app_tm	This application describes Test Module (TM) APIs.

app_3d

The app_3d application shows how to connect 3D stereo glasses (3DG) to BSA. Specific firmware (for the local Bluetooth controller) is required to use this feature.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

The following menus are available:

- Set 3DTV Idle Mode: Stops 3D information broadcast.
- Set 3DTV Master Mode: Starts 3D information broadcast.
- Set 3DTV Offset/Delay: Configures the shuttering parameters (left/right, open/close offsets) and the delay offset.
- Enable/Disable VSync Detection: Test functions to control VSync detection.
- Toggle 3D Proximity Pairing: Enable/Disable 3D bit of EIR to allow/disallow 3DG to perform Proximity
- Association.

Notes:

- The app_3d application is self-sufficient to control 3D stereo glasses devices (app_manager may not be executed).
- If other BSA applications (e.g. app_manager, app_av, etc.) are used, the app_3d application must be started at the end.

app_ag and app_hs

The app_ag and app_hs applications are used for mono headset connections. app_ag and app_hs are associated in this section based on operational codependence.

Both the server and applications must access the named socket file (bt-daemon-socket) which is created when the server is started.

The app_ag audio gateway is typically executed on a cellular application (such as a bridge between a cellular network and a headset).

The app_hs headset/handsfree is typically executed on a mono headset.

To initiate the first connection, place the stereo headset in pairing mode. A dedicated button is assigned for this purpose.

The connection can be established either from the AG or from the HS device. The AG application enables sending of the sco_ag_out.wav WAV file, which is located in the test_files/ag directory.

Both AG and HS applications can record received samples in a WAV file format based on options provided in a dedicated record menu. In this instance, the user must stop the file recording using the menu. This enables file readability before the audio is closed.

For demonstration purposes (such as when using an x86 computer), the AG application can read/write Pulse Code Modulation (PCM) samples from/to the local audio card (microphone/speaker) using the ALSA/asound driver/library.

Note: If the sco_ag_out.wav wave file is not available in the test_files/ag directory, the folder/file will need to be created.

app_av and app_avk

Both the server and applications must access the named socket file (bt-daemon-socket) which is created when the server is started.

The app_av and app_avk applications are used to stream advanced audio (such as stereo) between an AVS device to an AVK device. app_av and app_avk are associated in this section based on operational codependence.

- AVS devices are typically cellular-based or computer-based.
- AVK devices are typically stereo headset devices. In most cases, AVK devices require pairing.

AVS devices can send multimedia commands using the AV_RC protocol (these are included in the BSA AV and AVK profiles). The commands received by the AVS are used to control the stream (start/stop/pause, next/previous music, and so forth). The application receives these commands and performs the defined actions via the BSA AV APIs.

To initiate the first connection, place the stereo headset in pairing mode. A dedicated button is typically assigned for this purpose. The connection can be established either from the AV source or from the AVS device.

To play wave files in a play list (for AV), create a test_files/av directory, add one or several wave files, and start the app_av application. For demonstration purposes (such as when using an x86 computer), the AVK application can output the received stream to the computer speakers using the Advanced Linux Sound Architecture (ALSA)/asound driver/library.

To play the received stream on computer speakers, the AVK application must support the ALSA/asound driver.

app_ble

The app_ble application shows how to use BLE client/server module BSAs.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

The following menus are available:

- Discovery: Performs a discovery routine.
- Wake on BLE: Allows a remote BLE device to wake a local BT device host.
- For BLE client:
  - Register/Deregister Client App: Registers/deregisters a client application.
  - Connect: Opens a connection to a peer device.
  - Close: Closes a connection to a peer device.
  - Read/Write: Read/Write characteristic or characteristic descriptor from/to a peer device.
  - Service Discovery: Discover services on a peer device.
  - Register/Deregister Notification: Register notification to receive data from a peer device.
- For BLE server:
  - Create Service: Creates BLE server service.
  - Add Character: Creates characteristic on a service.
  - Start Service: Starts a service.
  - Register/Deregister Server App: Registers/deregisters a server application.

app_ble_blp

The app_ble_blp application is sample code for the BLE blood-pressure collector.

The following menus are available:

- Register Blood Pressure Collector: Registers a BLE client application.
- Discover Blood Pressure Sensor: Performs a discovery routine.
- Connect Blood Pressure Sensor: Opens a connection to a blood-pressure sensor.
- Search Blood Pressure Service: Searches for the blood-pressure service in a connected blood-pressure
- sensor device.
- Read Blood Pressure Feature: Reads the supported features of a blood-pressure sensor.
- Enable/Disable Blood pressure measurement: Enables and disables the receiving of blood-pressure
- measurements from a sensor.
- Start/Stop Intermediate Cuff Pressure measurement: Starts and stops the receiving of intermediate cuff-pressure values from a sensor while a measurement is in progress.

app_ble_cscc

The app_ble_cscc application is sample code for the BLE cycling speed and cadence collector.

The following menus are available:

- Register CSC Collector: Registers a BLE client application.
- Discover CSC Sensor: Performs a discovery routine.
- Connect CSC Sensor: Opens a connection to a cycling speed and cadence sensor device.
- Search Cycling Speed and Cadence Service: Searches for the cycling speed and cadence service from a connected cycling speed and cadence sensor device.

- Read CSC Sensor Feature: Reads the supported features of the server.
- Read CSC Sensor Location: Reads the physical location of the server when correctly fitted.
- Start/Stop Measure Data from Sensor: Starts and stops the receiving of CSC measurement data from the sensor.

app_ble_hrc

The app_ble_hrc application is the sample code for the BLE heart rate controller.

The following menus are available:

- Register Heart Rate Controller: Registers a BLE client application.
- Discover Heart Rate Sensor: Performs a discovery routine.
- Connect: Opens a connection to a heart rate sensor.
- Search Heart Rate Controller services.
- Monitor Heart Rate Sensor.

app_ble_htp

The app_ble_htp application is the sample code for the BLE health thermometer collector.

The following menus are available:

- Register Health Thermometer: Registers a BLE client application.
- Discover Health Thermometer Sensor: Performs a discovery routine.
- Connect Thermometer Sensor: Opens a connection to a thermometer sensor device.
- Search Thermometer Service: Searches for the thermometer service from a connected thermometer sensor device.
- Read type of Temperature Measurement: Reads the type of temperature measurement in relation to the location on the human body at which the temperature was measured.
- Enable/Disable Temperature measurement: Enables and disables the receiving of temperature measurements from the sensor.
- Start/Stop Intermediate temperature measurement: Starts and stops the receiving of intermediate temperature values from the sensor while a measurement is in progress.
- Read Measurement Interval: Reads the currently set measurement interval.
- Enable/Disable temperature measurement with interval: Enables and disables the controlling of temperature measurements with a measurement interval.

app_ble_pm

The app_ble_pm application is the sample code for the BLE proximity monitor.

The following menus are available:

- Register Proximity Monitor: Registers a BLE client application.
- Discover Proximity Reporter: Performs a discovery routine.

- Connect: Opens a connection to a proximity reporter.
- Search PM services.
- Configure Link Loss Alert/Immediate Alert level.

app_ble_rscc

The app_ble_rscc application is sample code for the BLE running speed and cadence collector.

The following menus are available:

- Register RSC Collector: Registers a BLE client application.
- Discover RSC Sensor: Performs a discovery routine.
- Connect RSC Sensor: Opens a connection to an RSC sensor device.
- Search Running Speed and Cadence Service: Searches for the RSC service from the connected RSC sensor device.
- Read RSC Sensor Feature: Reads the supported features of the server.
- Read RSC Sensor Location: Reads the physical location of the server when correctly fitted.
- Enable/Disable SC Control Point: Configures an SC control point for indications.
- Set Cumulative Value: Sets a new cumulative value. The new value is sent as a parameter following an op code.
- Start Sensor Calibration: Starts the calibration of the RSC sensor.
- Start/Stop Measure Data from Sensor: Starts and stops the receiving of RSC sensor measurement data.

app_ble_wifi

This is a sample application that makes it easier to set up a Wi-Fi direct connection between peers by exchanging Wi-Fi Direct information over BLE.

The following menus are available:

- Start WIFI RESPONDER: Start advertising Wi-Fi Information to create a Wi-Fi connection.
- Search WIFI RESPONDER: Discover a Wi-Fi responder.
- Start WIFI INITIATOR: Create a connection to a responder.
- Send WIFI data to responder: A notification message will be sent from a responder.
- Stop WIFI INITIATOR: Closes a connection with a Wi-Fi responder.

app_dg

The app_dg application shows how to use Data Gateway (DG) module BSAs.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

The following menus are available:

- Services Discovery (SPP): Performs a service discovery routine.

- DG Connect: Opens a connection to a peer device.
- DG Disconnect: Closes a connection to a peer device.
- DG Listen: Allocates a connection to a listen connection request from a peer device.
- DG Shutdown: Removes a connection that was allocated by the DG Listen command.
- Send a test file: Sends a test file to a DG connection.
- Toggle Mono/Multi connection mode: Toggles from the single- to multi-connection mode or from multi- to single-connection mode.

Because the app_dg application runs on a single thread, app_dg is busy when the user sends a file to a peer device. To avoid this condition, an additional thread should be created for file transmission.

app_fm

The app_fm application shows how to use the BSA FM APIs. The app can perform the following menu-driven functions:

- Enable/disable the FM radio.
- Tune the radio to a specific frequency
- Scan/search for a frequency
- Mute/Un-mute radio
- Control the audio volume
- Enable/disable RDS
- Set the radio geographical region.

app_ftc

The app_ftc application provides the Bluetooth FTP client (FTC) function.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

Ensure that the application side has been built with the following setting in app_ftc.txt:

```
BSA_FTC_INCLUDED = TRUE
```

Ensure that the server side has been built with the following setting in bte_server.txt:

```
BTA_FTC_INCLUDED = TRUE
```

app_fts

The app_fts application provides an example of how to use the file transfer server BSA API. This application can be run with an optional argument:

```
[-d pathname]
```

This argument stipulates the path and folder name of the test folder. By default this test folder is:

```
./fts_root
```

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

For testing, ensure that the folder used to start the server application contains a folder called /fts_root and that the permissions on this folder allow the remote device access for all operations.

Ensure that the application side has been built with the following setting in app_fts.txt:

```
BSA_FTS_INCLUDED = TRUE
```

Ensure that the server side has been built with the following setting in bte_server.txt:

```
BTA_FTS_INCLUDED = TRUE
```

app_hd

The app_hd application demonstrates Bluetooth HID device and remote control functionality. HID devices are typically keyboards, mice, remote controls, and unicast three-dimensional glasses.

Ensure that the application side has been built with the following setting in app_hd.txt.

```
BSA_HD_INCLUDED = TRUE
```

Ensure that the server side has been built with the following setting in bte_server.txt.

```
BTA_HD_INCLUDED = TRUE
```

app_headless

The app_headless application shows how the Headless mode is controlled. Headless mode is used to set the BT controller to allow peer devices to wake up the host. This feature is typically used to wake up (that is, exit Standby mode) DTV/STB products using a remote control.

To be allowed to wake up the host (by asserting a GPIO), a peer device must be added in the BT controller (by being saved to controller nonvolatile memory).

The following menus are available:

- Add Headless Device: Add BR/EDR HID device.
- Remove Headless Device: Remove BR/EDR HID device.
- Enable Headless: Sets the BT controller to Headless mode. This menu stops (kills) the BSA server.
- Add TvWakeUp Device: Similar to the Add Headless Device menu but with more control on HID reports.
- Remove TvWakeUp Device: Remove TvWakeUp Device.
- Add BLE Headless Device: Add BLE HID device. Similar to Add Headless Device but for BLE devices.
- Add BLE TvWakeUp Device: Similar to Add TvWakeUp Device but for BLE devices.

Note: Specific firmware is required to support this feature.

app_hh

The app_hh application is used to control the connection and the data stream to and from HIDs and exercise the three-dimensional Multicast Individual Poll (MIP) of the Bluetooth controller. HIDs are typically the keyboard, mouse, remote control, and unicast three-dimensional glasses.

HID devices typically employ the following guidelines:

- The keyboard requires pairing by entering the PIN code on the numeric keypad and pressing enter.
- The mouse does not require pairing.
- The remote control does not require pairing, except when using a remote control/keyboard combo.

An HID is typically neither discoverable (via inquiry scan) nor connectable (via page scan). To establish the first connection, place the subject devices in pairing mode. A dedicated button is typically used for this purpose.

Example: Press **Enter** and click **Start** for PS3 remote control.

Each HID has an HID descriptor which defines the reports that can be sent to and received from the device.

The application must read and save the HID descriptor in Nonvolatile Memory (NVM) during the first connection to the device. This descriptor is used to build and parse HID reports.

HIDs do not customarily enter page scan or inquiry scan mode, except when configured in pairing mode. HIDs automatically reconnect to the host when they need to send a report to the host (such as when there is mouse movement or a key is pressed).

app_hl

The app_hl application enables use of the Health Device Profile (HDP) module BSAs. The Health Level (HL) application also manages peer device data connections.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

The following menus are available:

- Device Class of Device (CoD) discovery (health): Discovers health peripheral devices based on class of devices.
- Device Service Discovery Protocol (SDP) discovery (health): Performs a device discovery, then an SDP on each device, and finally returns a list of the devices supporting a defined health service.
- SDP Query (Health): Used to retrieve health supported features for a specified device.
- Open Health Level (HL) Connection: Opens a data connection to an end point of a health peer device.
- Close HL Connection: Closes a data connection to a peer device.

To open HL data connections between HDP sink and source:

- Perform a CoD discovery or an SDP discovery to identify any HDP device. If any discoverable HDP devices are in range, they will be identified.
- Do an SDP query to obtain specific HDP device health service information.
- Open an HL connection to open a data connection.

app_manager

The app_manager addresses device management and security aspects such as responses to a PIN code reply.

The default PIN code used is 0000. The app_manager function also covers:

- Different pairing modes such as simple pairing, PIN code and link key.
- Discovery of remote devices and bonding, and establishing the first link to a device in order to create a link
- key (for authentication/encryption).
- Enabling peer devices to discover/connect/reconnect to the BSA. The process sets the local device to a
- discoverable and connectable mode. The app_manager must be running for this task to operate properly.
- Features not associated with a specific profile such as three-dimensional glasses in multicast legacy mode.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

For Set Top Box (STB) targets, the application executable is located in the build/mips folder. This command must be copied to and executed from the STB platform. This is in the same location used to execute the bsa_server.

app_mce

The app_mce application provides the Bluetooth MAP client (Message Client Equipment [MCE]) function.

Both the server and applications must access the named socket file (bt-daemon-socket) that is created when the server is started.

Ensure that the application side has been built with the following setting in app_mce.txt:

```
BSA_MCE_INCLUDED = TRUE.
```

Ensure that the server side has been built with the following setting in bte_server.txt:

```
BTA_MCE_INCLUDED = TRUE.
```

The following menus are available:

- Open Connection: Opens a connection to a MAP server on the peer device.
- Close Connection: Closes the currently open connection to the server.
- Start/Stop Notification Server: Starts/stops the message notification server on the client side.
- Register/Unregister Notification: Registers/unregisters for message-arrival notifications.
- Folder Listing: Retrieves a folder listing object from the current folder on the sever.
- Message Listing: Retrieves message listing objects.
- Get Message: Retrieves a specific message.
- Push Message: Pushes a message to a folder on the server.
- Update Inbox: Initiates an update of the server's inbox.
- Set Message Status: Modifies the read/delete status of a message.
- Set Folder: Navigates folders on the MAP server.
- Get MAS Instances: Retrieve information about MAS instances available on the MAP server.
- Get Server Instance Info: Retrieve user-readable information about a MAS instance from the MAP server.

app_opc

The app_opc application provides the Bluetooth (OPP) client (OPC) function.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

app_ops

The app_ops application provides the Bluetooth Object Push Profile (OPP) server (OPS) function.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

app_pan

The app_pan application provides the test framework for Bluetooth PAN functionality.

Both the server and applications must access the named socket file (bt-daemon-socket) that gets created when the server is started.

Setup:

Ensure that the application has been built with the following setting in the libbsa.txt file:

```
BSA_PAN_INCLUDED = TRUE
```

Ensure that the server has been built with the following setting in the bte_server.txt file:

```
BTA_PAN_INCLUDED = TRUE
```

app_pbc

The app_pbc application provides the Bluetooth PBAP client (PBC) function.

Both the server and applications must access the named socket file (bt-daemon-socket) that is created when the server is started.

Ensure that the application side has been built with the following setting in app_pbc.txt:

```
BSA_PBC_INCLUDED = TRUE
```

Ensure that the server side has been built with the following setting in bte_server.txt:

```
BTA_PBC_INCLUDED = TRUE
```

The following menus are available:

- Open Connection: Opens a connection to the PBAP server on the peer device.
- Get vCard: Gets the vCard entry for the specified object name (*.vcf).
- Get vCard listing: Gets the vCard listing object from the server based on the specified parameters.
- Get Phonebook: Gets the entire phone book object form the server.
- Set Folder: Sets the current folder in the virtual folder architecture on the server.
- Close Connection: Closes currently open connection to the server

app_pbs

The app_pbs application provides the Bluetooth Phone Book Access Profile (PBAP) server (PBS) function.

PBAP enables devices to exchange phone-book objects after a Bluetooth connection is established. Phonebook objects each represent information about a contact stored on a mobile phone.

Both the server and applications must access the named socket file (bt-daemon-socket) that was created when the server is started.

Ensure that the application side has been built with the following setting in app_pbs.txt

```
BSA_PBS_INCLUDED = TRUE
```

Ensure that the server side has been built with the following setting in bte_server.txt

```
BTA_PBS_INCLUDED = TRUE
```

app_sac

The app_sac application provides the Bluetooth SIM Access Profile (SAP) client function. SAP enables devices to access a GSM SIM card, a UICC card, or an R-UIM card via a Bluetooth link.

Ensure that the application side has been built with the following setting in app_sac.txt.

```
BSA_SAC_INCLUDED = TRUE
```

Ensure that the server side has been built with the following setting in bte_server.txt.

```
BTA_SAC_INCLUDED = TRUE
```

app_sc

The app_sc application provides the test framework for Bluetooth SAP server functionality.

Both the server and applications must access the named socket file (bt-daemon-socket) that gets created when the server is started.

Setup:

Ensure that the application is built with the following setting in the app_sc.txt file:

```
BSA_SC_INCLUDED = TRUE
```

Ensure that the server is built with the following setting in the bte_server.txt file:

```
BTA_SC_INCLUDED = TRUE
```

The following menus are available:

- Set Card Status: Sets the SIM card status.
- Set Reader Status: Sets the card reader status.
- Close Connection: Closes a currently open client connection.

app_switch

The app_switch application demonstrates the sequence of operations necessary to implement role switching between AV/AG and AVK/HF.

Ensure that the application side is built with the requirements specified for the app_av, app_ag, app_avk, and app_hf applications.

app_tm

The app_tm application shows how to use test modules with the BSA API.

Both the server and applications must access the named socket file (bt-daemon-socket) which is created when the server is started.

The application performs the following menu-driven routines:

- Enable/Disable Bluetooth Test Mode: This is used for Bluetooth qualification testing.
- Get server's and client's tasks and GKI buffers usage: This is used for debug/tuning purposes.
- Ping: Sends a ping request to the local BSA server. This is used for debug purposes.
- Send: Sends vendor-specific commands to the local Bluetooth device.
- Enable /disable server traces.
- Register a call back to receive vendor specific events.
- Read firmware and BSA version information.
- Send HCI LE test commands to the controller.

Broadcom

Web: www.broadcom.com

Irvine, CA 92617

BSA-SERVER-SWUM101-R March 08, 2016