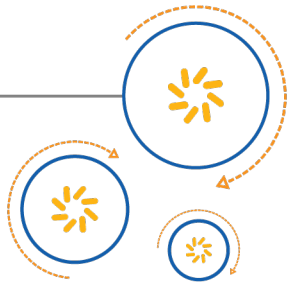
Qualcomm Technologies International, Ltd.



Guide to Qualcomm Bluetooth SDK Libraries

User Guide

80-CT435-1 Rev. AJ

October 18, 2017

Revision history

Revision	Date	Description
1	JUL 2010	Initial release. Alternative document number CS-00207478-UG.
2	JUL 2011	Updated to latest CSR™ style
3	JAN 2012	Updated to latest CSR style
4	JAN 2013	Updated for ADK 2.5
5	APR 2014	Updated to latest CSR style
6	APR 2016	Updated to conform to QTI standards; no technical content was changed in this document revision.
7	APR 2017	Technical updates to reflect latest ADK releases
AH	APR 2017	Removed watermark. Updated Document Reference to use agile reference.
AJ	OCT 2017	Added to Content Management System. No change to technical content.

Contents

Tables

Figures

1 Overview of SDK and ADK libraries

The SDK and ADK libraries sit between the application and the Qualcomm® BlueCore™ technology firmware.

It is not necessary to know all the function calls in every library. This information is available in the VM and Native Reference documentation provided in the xIDE on-line help. However, it is worthwhile taking the time to gain an overall feel of the different libraries and the range of functions available.

An understanding of the libraries gives an insight into what is supported by the firmware and therefore what it is possible to achieve in a particular application.

It also helps software engineers new to Bluetooth to interpret the application code or example code supplied.

Three classes of library can be considered:

- Foundation Libraries
- Support Libraries
- Profile Libraries

Between them, these libraries provide software engineers with a comprehensive range of functions that can be used to create Bluetooth applications to run on BlueCore ICs.

1.1 SDK and ADK foundation libraries

The foundation libraries implement the basic features exposed by the BlueCore firmware and handle communication with the lower levels of the Bluetooth stack, largely hiding the complexity from the developer.

The source code for the foundation libraries is not provided in the SDKs and ADKs.

The libraries that fall into this category are:

Table 1-1 Standard C 'foundation' libraries

Standard C 'Foundation' libraries	
`assert`	Allows assertions to be activated/deactivated using the `NDEBUG` macro identifier, to aid documentation and debugging of code
`CapacitiveSensor`	Functions to configure the touch sensor hardware
`ctype`	Implements character handling functions in a non system dependent way
`iso646`	Defines macros for alternative representation of logical operators
`lcd`	Functions to configure the LCD hardware

Table 1-1 Standard C 'foundation' libraries (cont.)

Standard C 'Foundation' libraries	
limits	Defines the maximum/minimum limits of basic data types
memory	Implements certain dynamic memory allocation/deallocation operators
stdarg	Defines macros used to get the arguments in a function when the number of arguments is not known
stdbool	Defines macros used to define and undefine Boolean types and values
stddef	Defines various standard types and macros
stdio	Defines ANSI Output functions used for outputting debug printfs
stdlib	Library definitions for various standard ANSI C utilities
string	Defines various standard string handling functions
NOTE SDK versions of the standard libraries are cut-down versions that restrict the functions available to those supported by the embedded environment of the BlueCore firmware.	

Table 1-2 Foundation libraries

QTIL Foundation libraries	
adc	Functions used to obtain voltage readings from the Analogue to Digital Converter (ADC) hardware.
boot	Used to control BlueCore boot modes on unified firmware
charger	Functions to configure the on-chip battery charger present on some BlueCore chips
codec_	Functions to control the audio CODEC
energy	Functions used to enable and disable energy estimation on a given SCO connection
file	Functions used to access the read-only file-system
font	Provides support for the font subsystem
host	Function used to send a message to the host
i2c	Allows transfer of data across the I2C interface
inquiry	Functions to configure the Bluetooth Inquiry procedure
kalimba	Functions used to control the Digital Signal Processor (DSP) on BlueCore Multimedia chips
led	Functions to control the LED hardware present on some BlueCore chips
link	Enables link key snooping for dual boot devices
message	Functions used to control message passing
micbias	Functions to control the microphone bias hardware
panic	Functions that can be used to panic the application
pio	Functions used to access BlueCore Input/Output lines
ps	Provides access to the persistent key store on Bluecore
psu	Functions to control the power supply hardware
ringtone_notes	Macros used to generate tones
sink	Functions used to output data on 8-bit streams

Table 1-2 Foundation libraries (cont.)

QTIL Foundation libraries	
source	Functions used to handle data arriving on 8-bit streams
sram	Functions to map SRAM into VM memory space
status	Function that queries the value of specified status fields
stream	Functions for efficiently processing streams of 8-bit data
test	Functions used to enter BlueCore test modes
transform	Functions used to process data flowing between source and sink
usb	Functions used in the control of USB end points
util	Utility functions used to perform a number of frequently required tasks
vm	Functions providing low level access to BlueCore firmware and hardware

Using foundation libraries

Both the standard C and QTIL Foundation libraries are automatically available to the linker when code is compiled in xIDE.

In addition to the foundation libraries listed, two other libraries that can be loosely classed as foundation libraries are provided for convenience:

- QTIL: including the header file for this library (`CSR.h`) in your source code pulls in all the SDK support and application libraries.

  NOTE This saves the developer having to specify individual libraries but increases the time taken to compile the code.

- CSRtypes: this library declares all the QTIL specific types used in the support and application libraries. Any other header file that requires the `typedef`s in the `CSRtypes` library itself includes `CSRtypes.h`.

The foundation libraries are further documented in the VM and Native Reference Guide provided in the xIDE on-line help.

1.2 SDK and ADK support libraries

These comprise a set of libraries that help the `connection` library to implement RFCOMM, L2CAP and SCO connections in a simple manner.

They also facilitate other features such as interpretation and handling of other raw data available from the foundation libraries, for example, battery status readings.

Support libraries	
audio	CSR_voice_prompts_plugin
audio_plugin_common	display
audio_plugin_if	display_example_plugin
batt_rep	display_plugin_cns10010
power	display_plugin_if

Support libraries	
bdaddr	fm_plugin_if
codec	fm_rx_api
connection	fm_rx_plugin
CSR_a2dp_decoder_common_plugin	kalimba_standard_messages
CSR_a2dp_encoder_common_plugin	md5
CSR_ag_audio_plugin	pblock
CSR_common_example_plugin	pio_common
CSR_cvc_common_plugin	print
CSR_dut_audio_plugin	obex_parse
CSR_fm_audio_plugin	region
CSR_simple_text_to_speech_plugin	sdp_parse
CSR_speech_recognition_plugin	service
CSR_subwoofer_plugin	usb_device_class
CSR_tone_plugin	-

The header files (.h files) and where relevant the source code (.c files) for these libraries can be found in:

<SDK installation folder>\src\lib

To use a support library its header file must be specified by the #include directive in the application source file and where appropriate the library must be listed in the xIDE Project Properties, see chapter Using Bluetooth libraries in xIDE.

1.3 SDK and ADK profile libraries

These libraries implement the Bluetooth Profiles as defined in the Bluetooth specifications (see http://www.bluetooth.org)

The profile libraries ultimately interface with the connection library or with another profile library, which in turn interfaces with the BlueCore firmware.

The libraries in this category are:

A2DP, see the *A2DP Library User Guide*	gatt
aghp	mapc
avrcp	md5
batt_rep	obex
hfp, see the *HFP Library User Guide*	oppc
hid	pbapc
kalimba_standard_message	sppc,spp and spp_common

The header files (.h files) and source code (.c files) for these libraries are found in:

```
C:\<SDK Install directory>\src\lib
```

To use a profile library its header file must be specified by the `#include` directive in the application source file and the library must be listed in the xIDE Project Properties, see Using Bluetooth libraries in xIDE.

NOTE The properties of supplied projects automatically list the libraries used by the code.

2 SDK and ADK library architecture and dependency

The library functions supplied with an SDK or ADK expose the BlueCore stack to the application code in structured layers.

The figure below gives a conceptual representation of the library hierarchy.

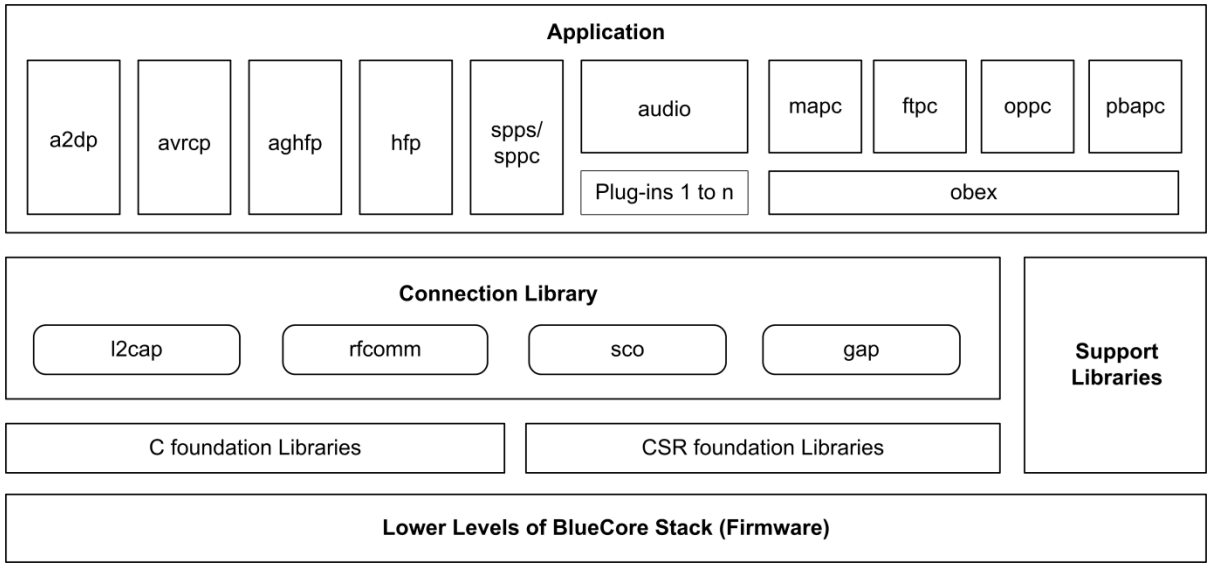


Figure 2-1 Library architecture

2.1 SDK and ADK library interaction

To better understand how the libraries interact, it is useful to consider some applications and how functions from other libraries can be used to extend functionality.

A simple headset application would use functions from the libraries highlighted in the figure below.

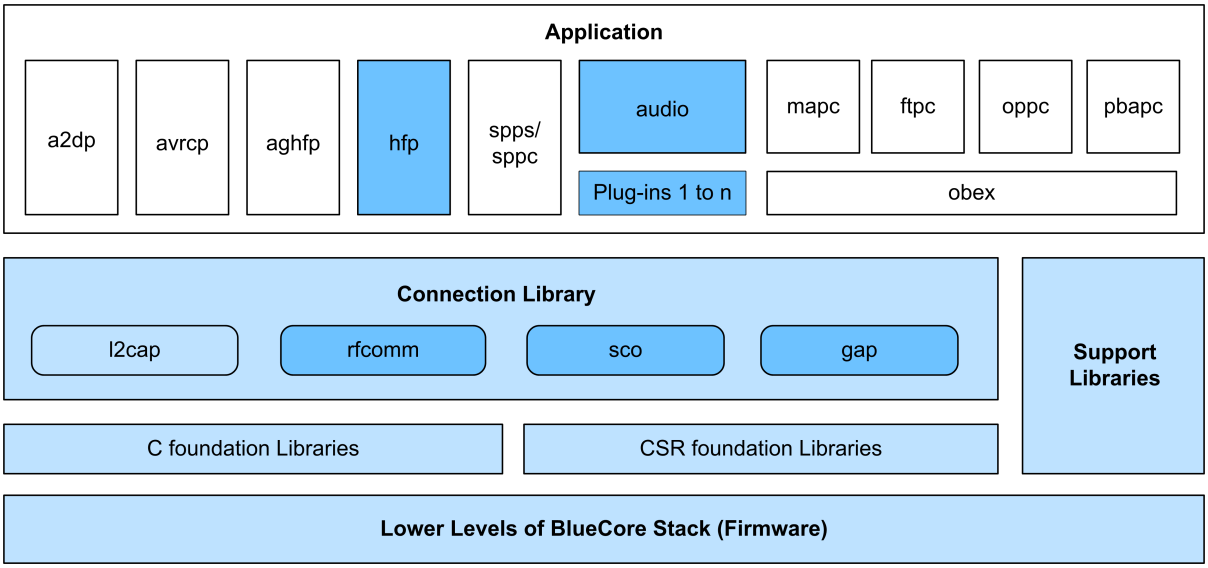


Figure 2-2 Libraries used to implement a simple headset application

2.1.1 Adding functionality to a BlueCore application using Bluetooth libraries

The example and reference application code provided in Bluetooth is intended as a starting point for engineers developing their own applications.

NOTE Application specific SDKs provide a fully functional Bluetooth design.

For example the introduction of code to monitor the battery status would probably be required in most commercial applications and would need to be added if developing an application from example plug-ins.

To implement battery monitoring the software engineer would need to add the necessary code to initiate periodic battery readings and to handle responses, using the functions from the battery library.

NOTE Support information required to use library functions, for example, function types, parameters passed, and so on, can be found in *VM and Native Reference Guide*, which is part of the xIDE on-line Help documentation.

The header file would then need to be specified by the `#include` directive in the source code and the battery library added to the **Project Properties** before the code could be compiled and linked in xIDE.

Additional Bluetooth functionality can be implemented, in a similar way, by using functions from other profile libraries.

For example, the additional functionality in a stereo application requires use of the `a2dp`, `avrcp` and `connection` libraries:

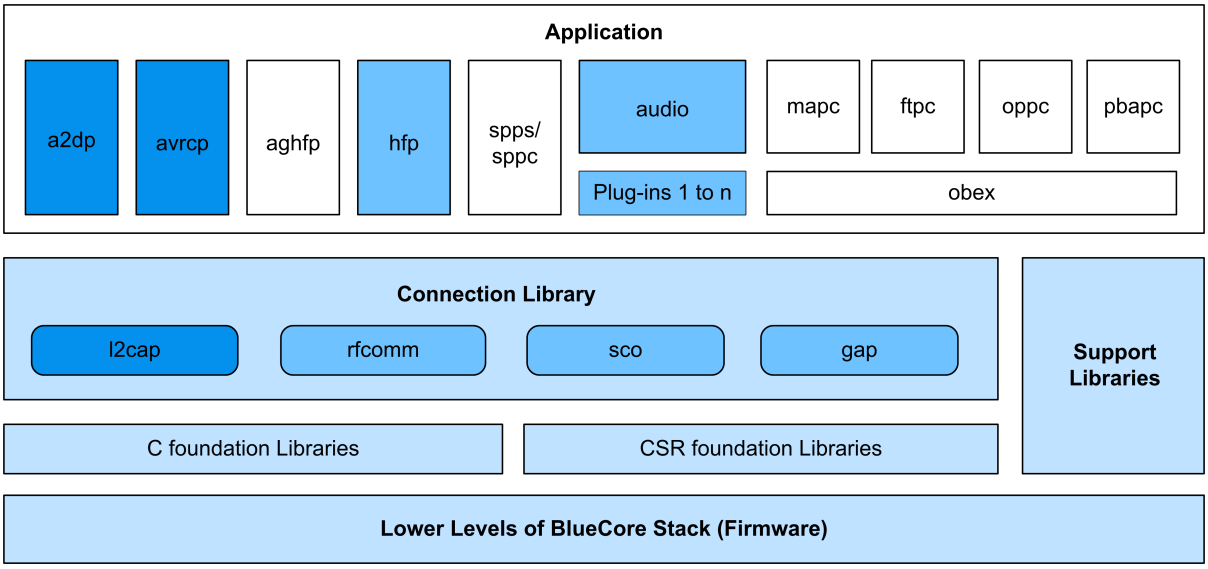


Figure 2-3 Libraries used to implement a typical stereo headset application

3 Using Bluetooth libraries in xIDE

When using Profile or Support library functions in application code it is important to ensure that the appropriate header file is included with an `#include` statement and that non-foundation libraries are added to the Project Properties in xIDE.

NOTE When example or reference application code is loaded into xIDE the libraries used within the code will already be part of the **Project Properties**.

If the library header is not specified by the #include directive in the source code, the compiler cannot correctly compile calls to library functions. If the library is not listed in the **Project Properties** the linker cannot include the actual code for these functions from the `<lib name>.a` file.

3.1 How to include a library header file

The header files declare related groups of functions and variables. To make use of a library's functions in a source file the library header must be included in that source file. for example:

```
#include <lib name>.h
```

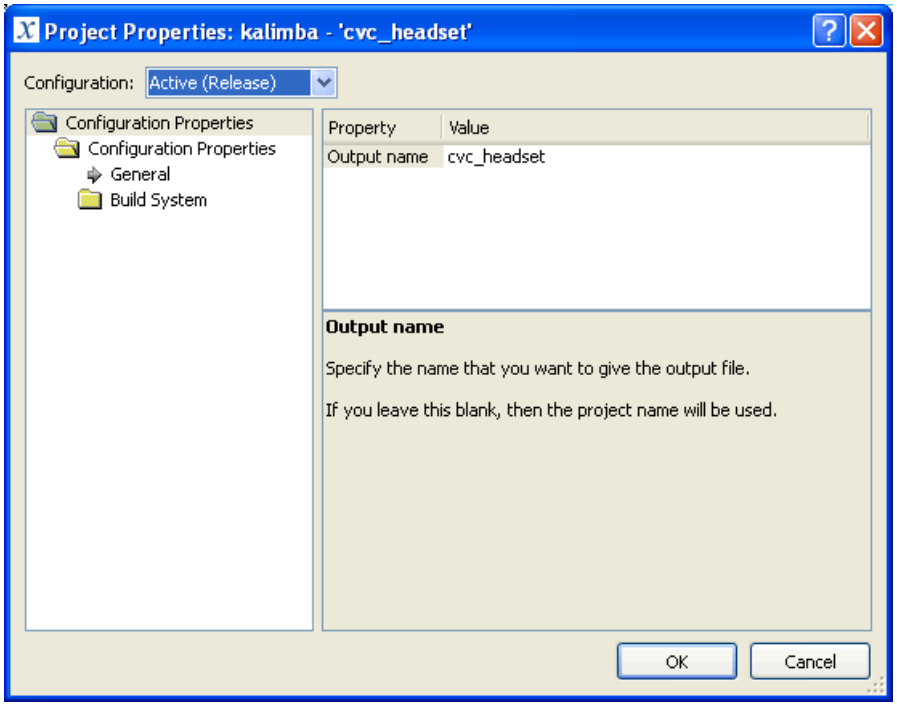
It is good practice for include statements to be placed at the beginning of source files in which the functions they declare are used.

3.2 How to add a library to Project Properties

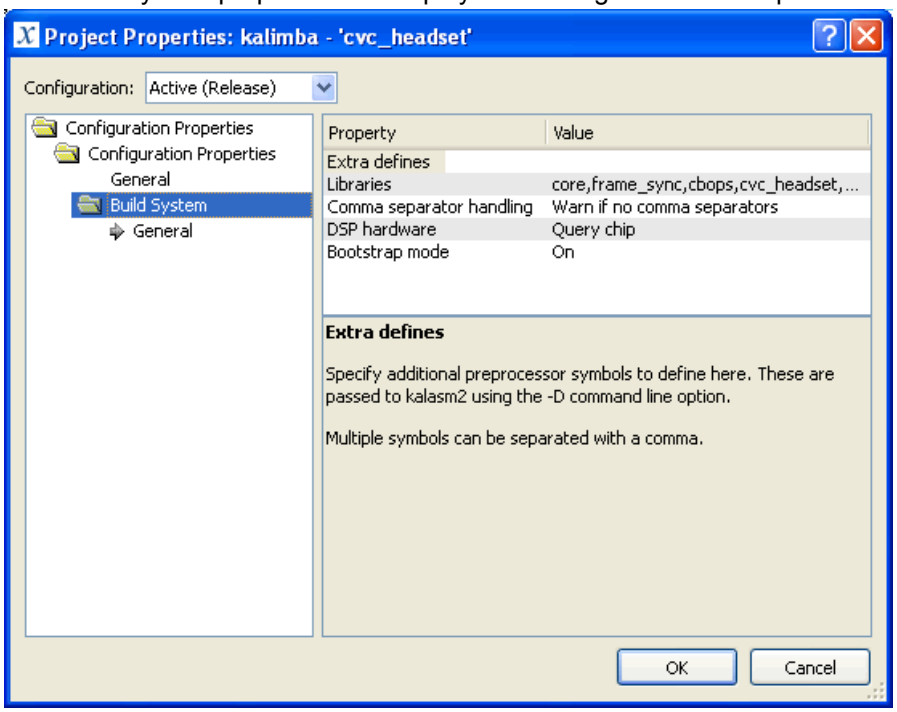
When library functions are used in code, it is important that the library is present in the list of libraries in the xIDE **Project Properties**. This enables the compiler to find the code when linking.

To add a library to the Project Properties

1. Select **Properties** from the xIDE **Project** menu.
   The **Project Properties** window appears:



2. Click on the **Build System** folder.
   The Build System properties are displayed in the right-hand workspace:



3. Click on the **Libraries** row to activate the **Value** field.
4. When the required libraries have been added, click **OK** to set the properties for the project.

3.3 How to build/rebuild libraries

The Bluetooth libraries are built as part of the Bluetooth installation process and it is not generally necessary to rebuild the libraries.

However, if the default option to build libraries was unchecked during installation or the source code within a library has been amended (although this is not recommended procedure) they can be rebuilt from the Windows **Start** menu.

To rebuild libraries go to `Start/Programs/<SDK Name>/Rebuild` and select the libraries you want to rebuild:

4 Bluetooth Profile and Support library reference

A detailed listing of the data structures, definitions, variables and message structures can be found in the VM and Native Reference Guide, which is part of the xIDE online Help documentation.

In addition, reading the header files themselves (which are comprehensively and clearly commented) helps in understanding the functions provided in each library.

General

The naming conventions used in Bluetooth are designed to help engineers readily identify functions, their associated messages and the library they are from.

For example, functions and their associated messages are prefixed by the name of the library and have a structured naming convention.

For example, the function `A2dpMediaOpenRequest` opens a local Stream End Point (SEP) and causes the confirmation message `A2DP_MEDIA_OPEN_CFM_T` to be returned.

When a remote device opens a local SEP the application receives the following indication message `A2DP_MEDIA_OPEN_IND_T`.

When writing an application using Bluetooth, QTIL recommends that the primary consideration is the profile(s) to be implemented, rather than any other requirements such as the type of connection.

Using the appropriate profile libraries allow implementation of the required connections as part of the profile library's function calls to the `connection` library.

The following table lists the Bluetooth Profile libraries.

Table 4-1 Bluetooth profile libraries

Library	Description
`a2dp`	Interface to the Advanced Audio Distribution Profile (A2DP) library. See the *A2DP Library User Guide*.
`agfhp`	Interface to the Audio Gateway Hands Free Profile library.
`audio`	Support library that facilitates audio connections. See the *Introduction to the Audio Library*.
`avrcp`	Interface to the Audio Video Remote Control Profile library.
`Batt_rep`	Interface to the Battery Reporter service library.

Table 4-1 Bluetooth profile libraries (cont.)

Library	Description
BdAdrr	Helper routines for Bluetooth addresses.
codec	This library implements the functionality required to configure an internal or external stereo codec for use.
connectio	Used to facilitate L2CAP, RFCOMM, SCO connections and GAP functionality. See the *Connection Library RFCOMM API Application Note*.
display	Plugin based library structure for implementing displays.
fm_rx	Plugin based library structure for implementing fm receive functionality.
gatt	Interface to the Generic Attribute Profile (GATT).
hdp	Interface to the Health Device Profile (HDP)
hfp	Interface to the Hands Free Profile (HFP) library. See the *HFP Library User Guide*
hid	Interface to the Human Interface Device Profile
Kalimba_standard_messages	The messages passed between the kalimba libraries and the VM application
mapc	Interface to the Message Access Profile(MAP) client library
md5	Interface to the md5 library
obex	Interface to the Generic Object Exchange Profile (GOEP) library
oppc	Interface to the Object Push Profile (OPP) Client library
pbapc	Interface to Phonebook Access (PBAP) Client library
piolib	Allows more than one task to receive MESSAGE_PIO_CHANGED messages
power	Take battery readings and return them in milliVolts
print	Debug print functions
region	Processes regions of uint8 memory
service	Library to search service records for regions of interest
spp	Interface to the Serial Port Profile (SPP) library

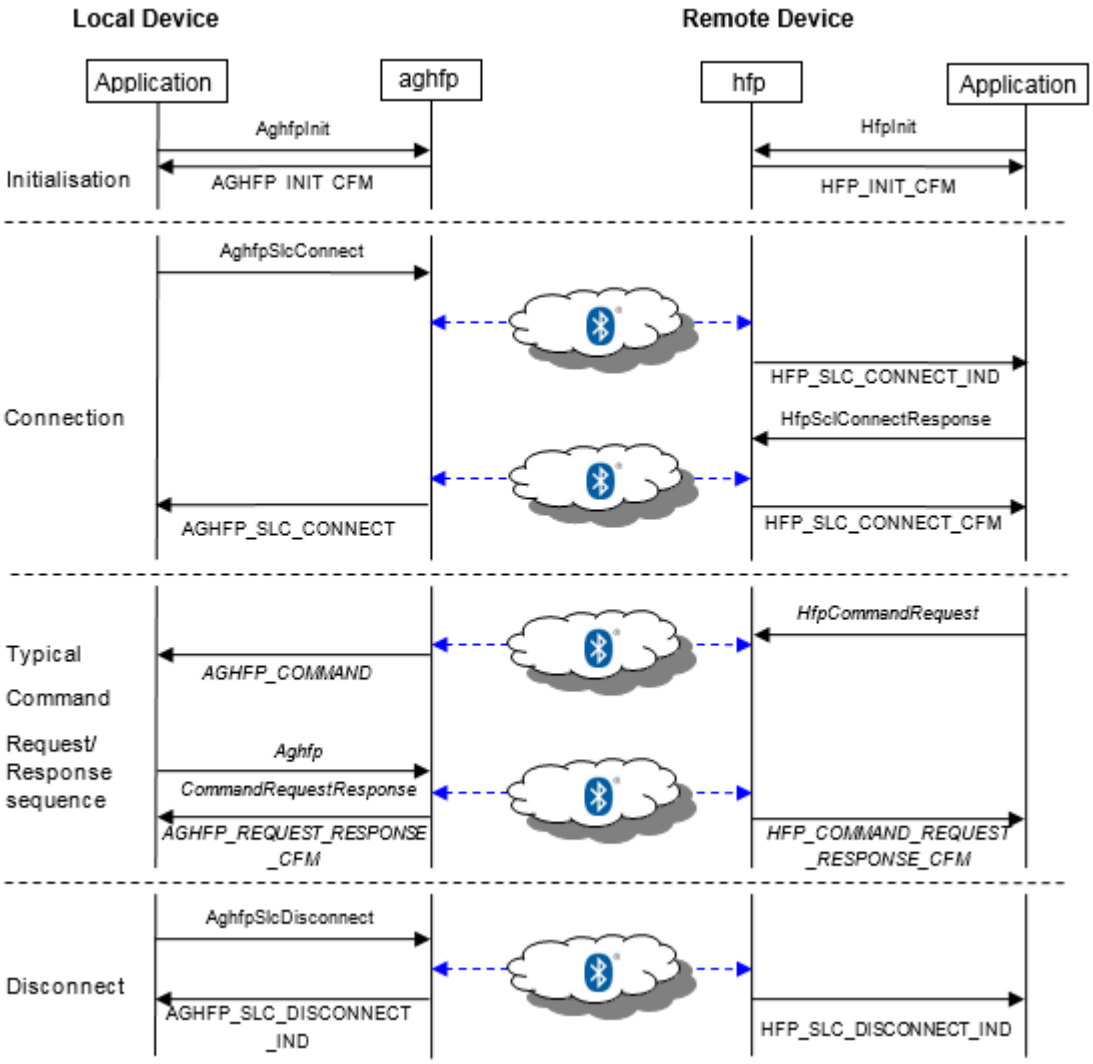
4.1 aghfp library

Interface to the Audio Gateway Hands Free (AGHFP) Profile.

Purpose

The Audio Gateway Hands Free Profile defines functions for an audio gateway device (fror example, a mobile phone) connecting and interacting with a hands free device (e.g. an hfp headset).

Typical message sequence chart



4.2 audio library

The audio library is described in *Introduction to the Audio Library*.

4.3 avrcp library

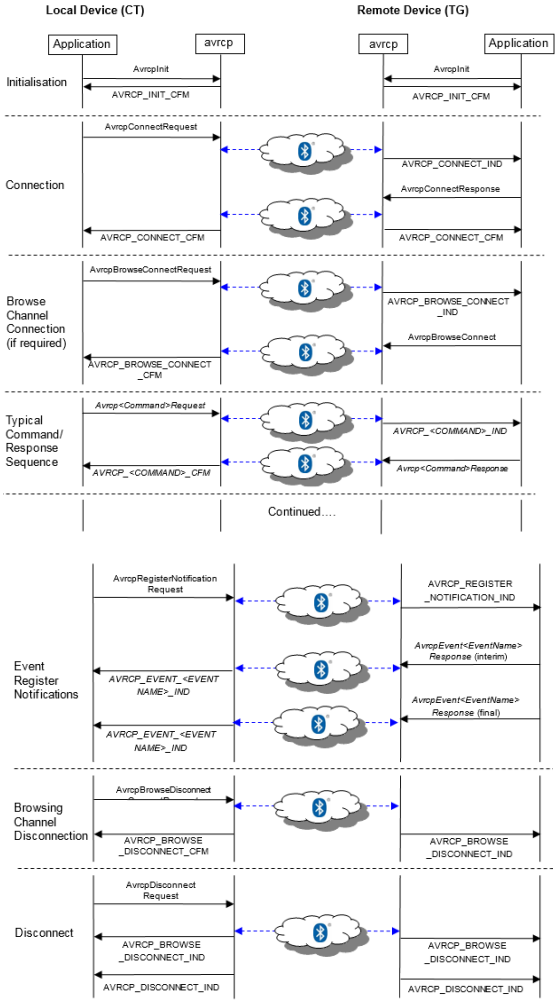
Interface to the Audio Video Remote Control Profile (AVRCP) library.

Purpose

This Profile library implements the Audio Video Remote Control Profile v1.4. It permits one device known as the controller (CT) to send dedicated user actions to another device known as the target (TG).

The library exposes a functional downstream API and a message-based upstream API.

Typical message sequence chart



NOTE *Italic* labels in the sequence chart indicate generic functions and messages. See the VM and Native Reference Guide in the xIDE on-line help for a list of actual Command Requests.

An example would be the `AvrcpPassthrough` command/response (for a pass through request to be sent to the target device).

4.4 batt_rep library

Interface to the Battery Reporter service library.

Purpose

This library implements the Battery Service (BAS). This is a GATT based service and designed primarily for use over Bluetooth low energy.

4.5 Bdaddr library

Interface to allow Bluetooth address handling.

Purpose

This library defines various functions that facilitate the comparison and handling of Bluetooth addresses.

Example

This section gives a typical example of the type of function included in this library.

Where a variable is used to store a Bluetooth address it may be desirable to set it to a value that can be readily identified as not being a real Bluetooth address, for example when deleting a pairing.

This can be done using the function `void BdaddrSetZero (bdaddr*)` which is declared in the `bdaddr.h` header file.

This function sets the value of the variable specified to zero, which is always an invalid Bluetooth address.

4.6 codec library

Interface to the `codec` library.

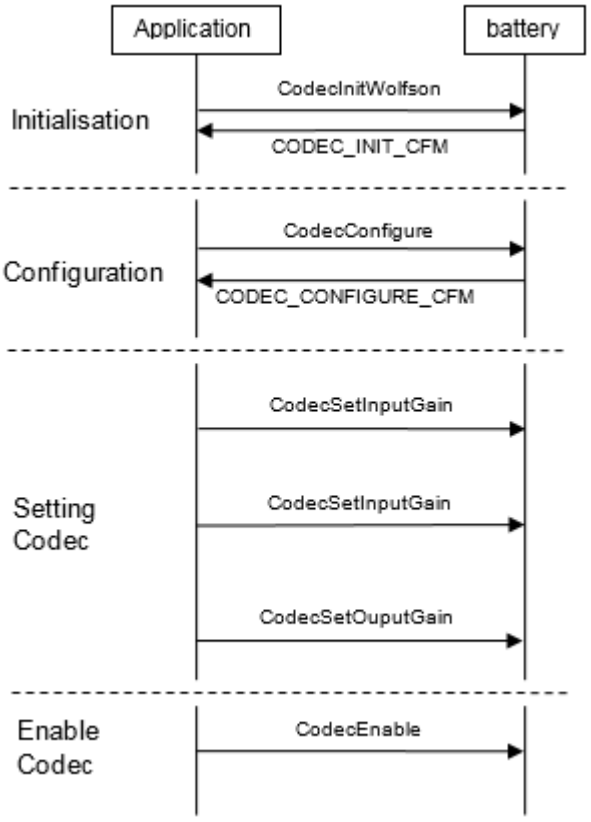
Purpose

This library implements the functionality required to use an audio stereo codec.

It provides support for initializing, configuring and using a particular codec.

Typical message sequence chart

The message sequence chart shows a typical sequence of initialization and set up for an external Wolfson codec.



4.7 connection library

Interface to the connection library. See the *Connection Library RFCOMM API Application Note*.

Purpose

The library provides a comprehensive set of functions and associated response messages.

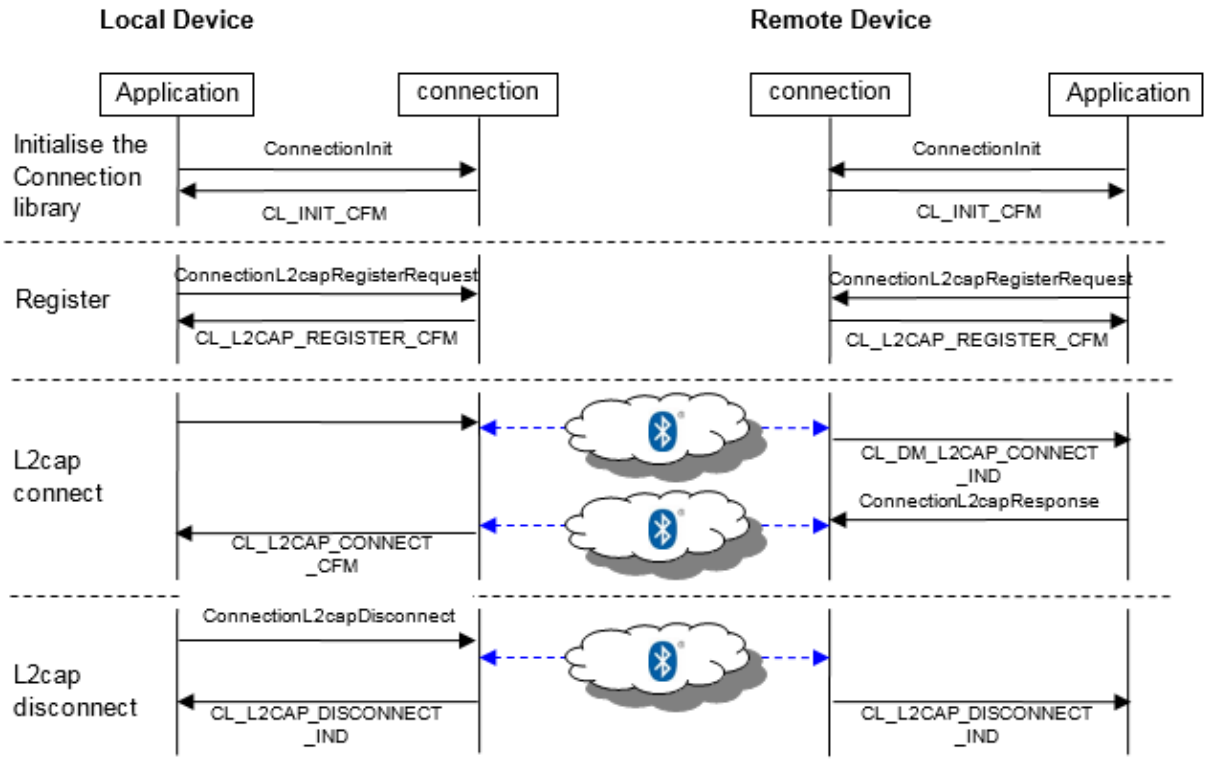
The services provided include support for RFCOMM, L2CAP and SCO connections, SDP search primitives and access to the Device Manager (DM) layer that exposes the Host Controller Interface (HCI) layer functionality.

NOTE The naming conventions used, help to identify the purpose of each function.

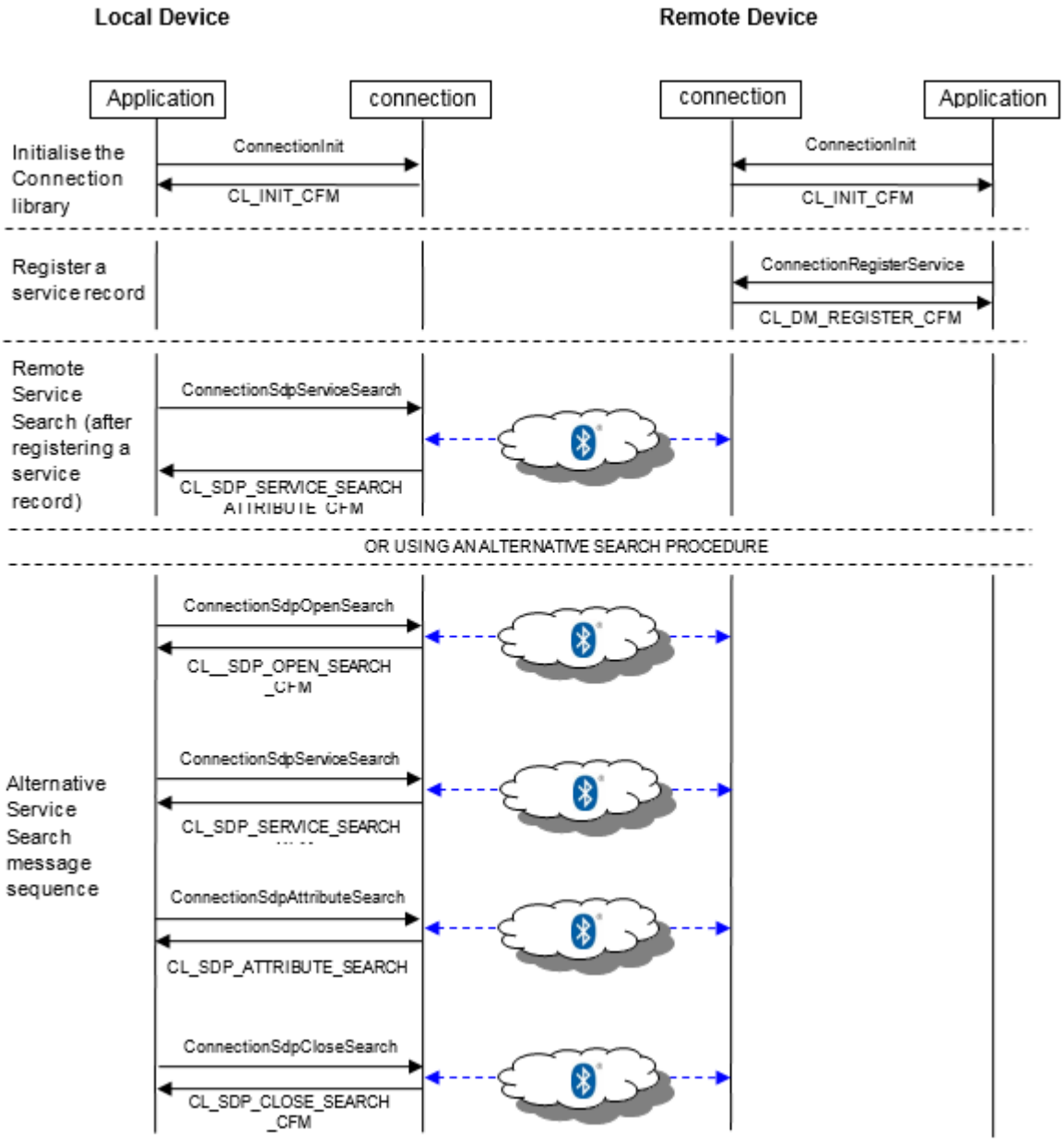
The xIDE on-line C Reference Guide gives any additional information needed to use connection library functions in application code.

Typical message sequence chart

The charts below illustrate examples of typical message sequences using the connection library. They are illustrative only and do not represent actual implementations of an application.



NOTE The message sequence chart above shows a profile interaction with the `connection` library on both the local and remote device.



When a service record has been registered, responses to service search requests are handled by BlueStack and are not seen by the Application.

4.8 display library

Plugin based library structure for implementing displays.

Purpose

The display set of libraries has a plugin based structure so the display hardware can be changed without affecting the operation of the main application, for example a soundbar. The following plugins are provided:

- `display`: Main interface to the display.
- `display_plugin_if`: Message interface used for the display plugin.
- `display_example_plugin`: An example plugin framework (uses the debug print channel).
- `display_plugin_cns10010`: An example plugin for use with the QTIL CNS10010 development board.

The application should implement the display using the Display library and initialize it to the required plugin with the `DisplayInit()` function.

Different display hardware can be supported by creating new plugins.

4.9 fm_rx library

Plugin based library structure for implementing FM receive functionality.

Purpose

The FM set of libraries has a plugin based structure so the FM Receive hardware can be changed without affecting the operation of the main application, for example a headset. The following plugins are provided:

- `fm_rx_api`: Main interface to the FM Receive module.
- `fm_plugin_if`: Message interface used for the FM Rx plugin.
- `fm_rx_plugin`: An example plugin for use with the FM Rx device fitted to the QTIL CNS10010 development board.

The application should implement the FM Rx using the `fm_rx_api` library and initialize it to the required plugin with the `fmRxInit()` function.

Different FM hardware can be supported by creating new plugins.

4.10 gatt library

Interface to the Generic Attribute Profile (GATT).

Purpose

The profile describes a use case, roles and general behaviours based on the GATT functionality. See the *Bluetooth Core Specification Version 4.0* or later.

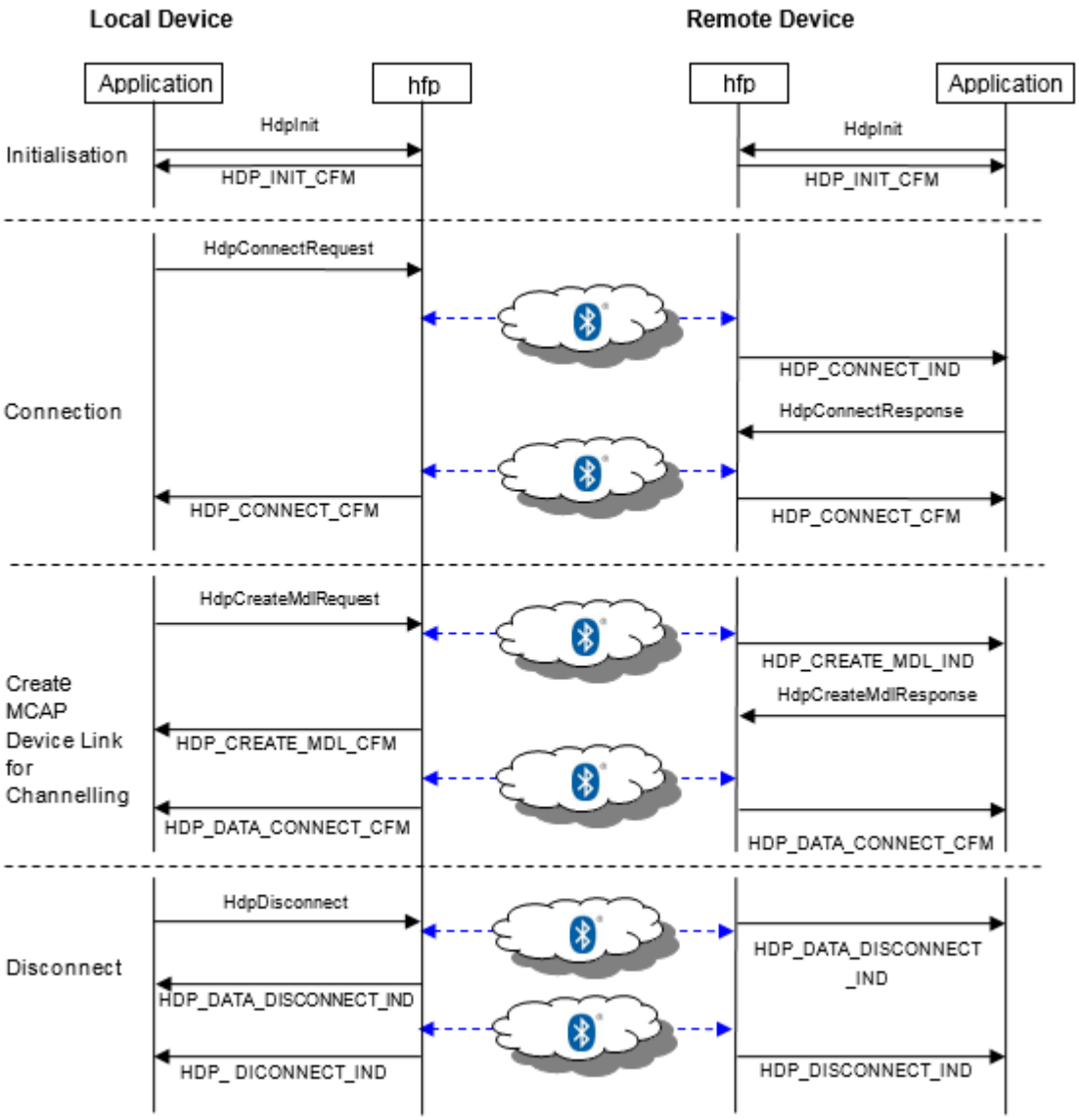
4.11 hdp library

Interface to the Health Device Profile library.

Purpose

This library implements the Health Device Profile (HDP) and the Multi-Channel Adaptation Protocol (MCAP), enabling interoperability between medical, healthcare and fitness applications from different vendors.

Message sequence chart



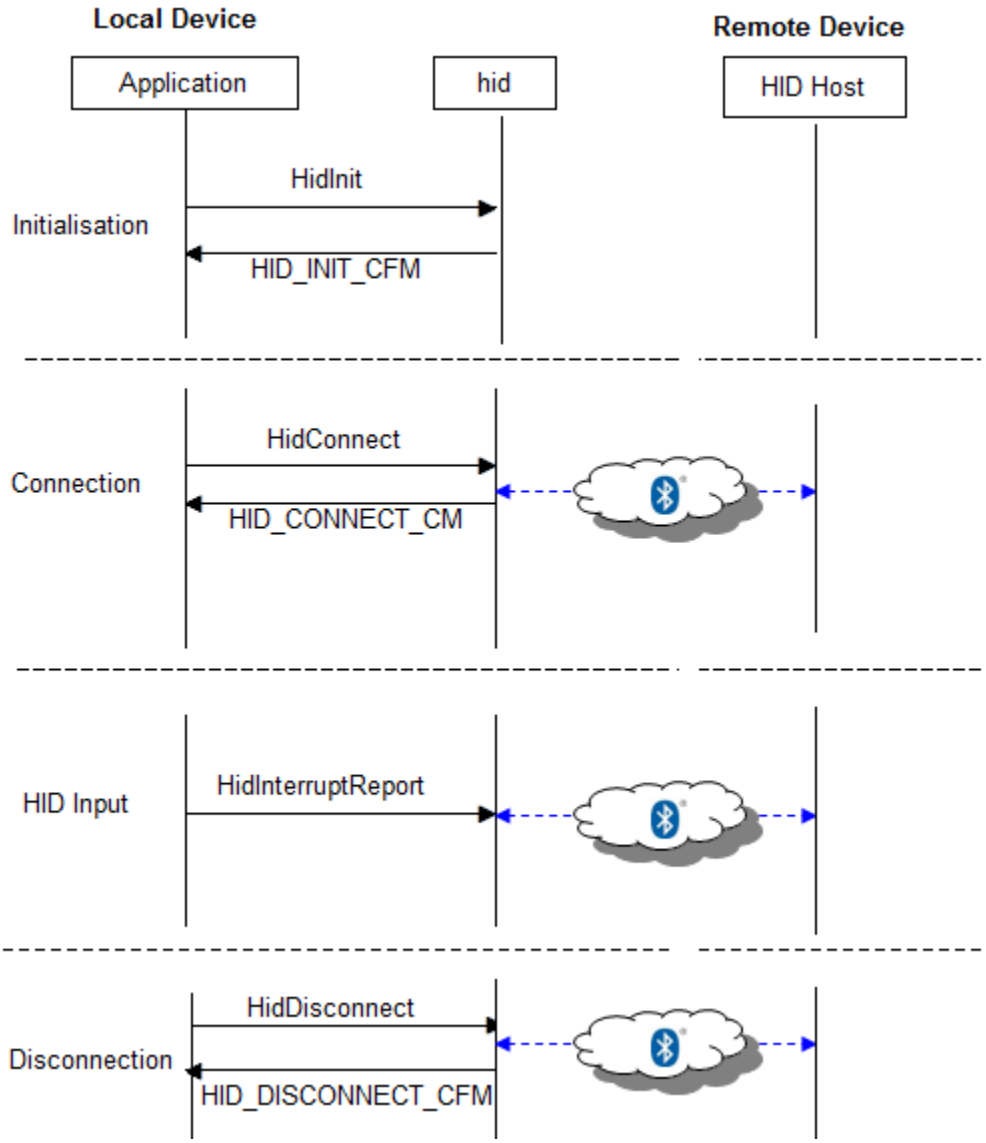
4.12 hid library

Interface to the Human Interface Device (HID) library.

Purpose

This library implements the Human Interface Device Specification for a Bluetooth HID Device. It provides a service to input and output data to and from a Bluetooth HID host.

Typical message sequence chart



4.13 kalimba_standard_messages library

This is not a library (as it does not define any functions) but is a header file for Qualcomm® Kalimba™ messages.

Purpose

This header file defines the message IDs used by Bluetooth libraries to communicate with the Digital Signal Processor (DSP) in BlueCore Multimedia chips.

General

Users wanting to define their own kalimba messages should restrict the message ID to the range `0x0` to `0x6fff`. Values over this are reserved for `kalimba_standard_messages`. See the xIDE on-line DSP Reference Guide for more details.

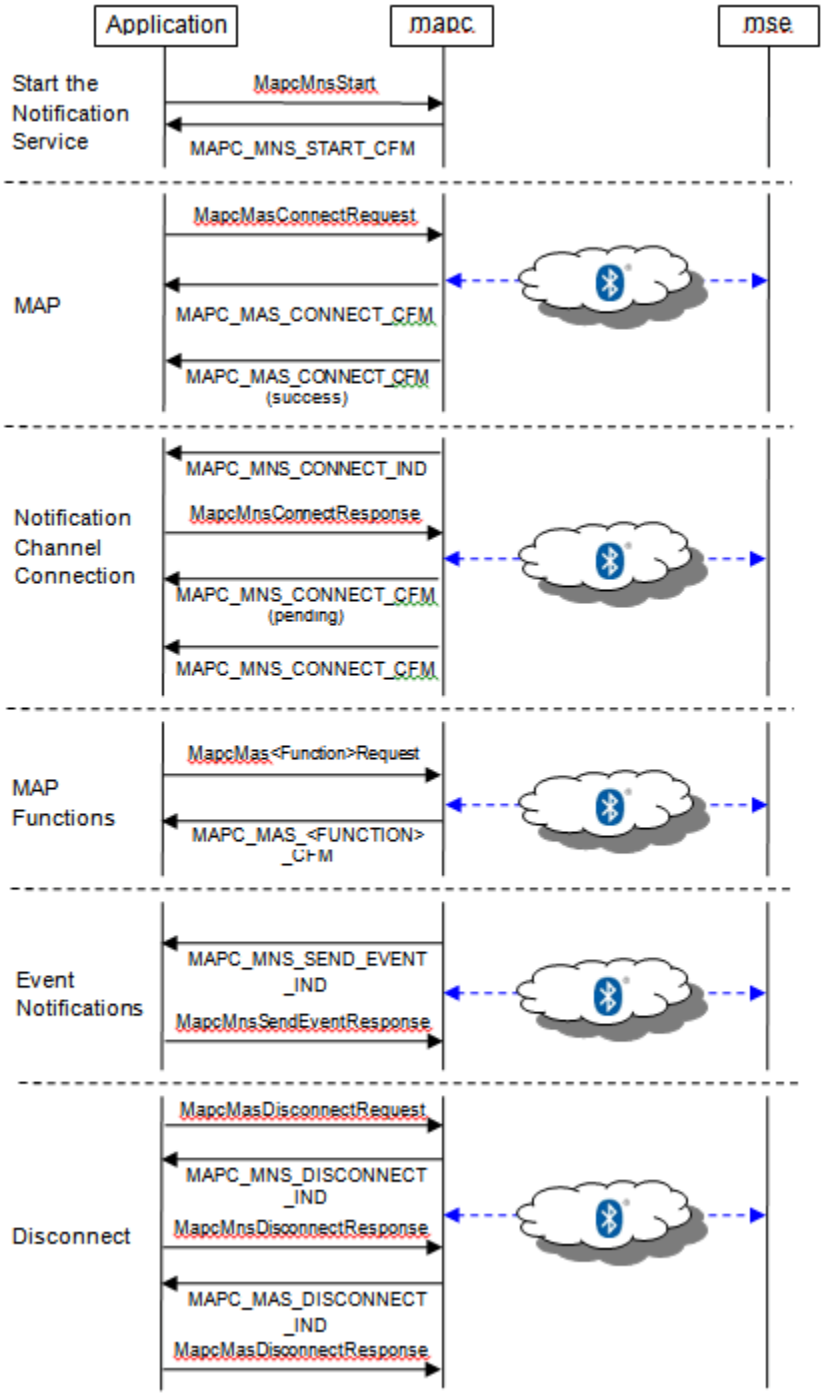
4.14 mapc library

Interface to the Message Access Profile (MAP) library.

Purpose

This library facilitates exchange of short messages between devices. The primary example is between hands free devices such as car kits and mobile phones.

Typical message sequence chart



4.15 md5 library

Interface to the `md5` library.

Purpose

This library implements MD5 message-digest based on the RSA reference implementation in RFC-1321. MD5 is a widely used cryptographic hash function used to check the integrity of data blocks.

The library has been added to support authentication in PBAP. It is too slow to be used to validate large datasets.

See the xIDE on-line VM and Native Reference Guide for more details.

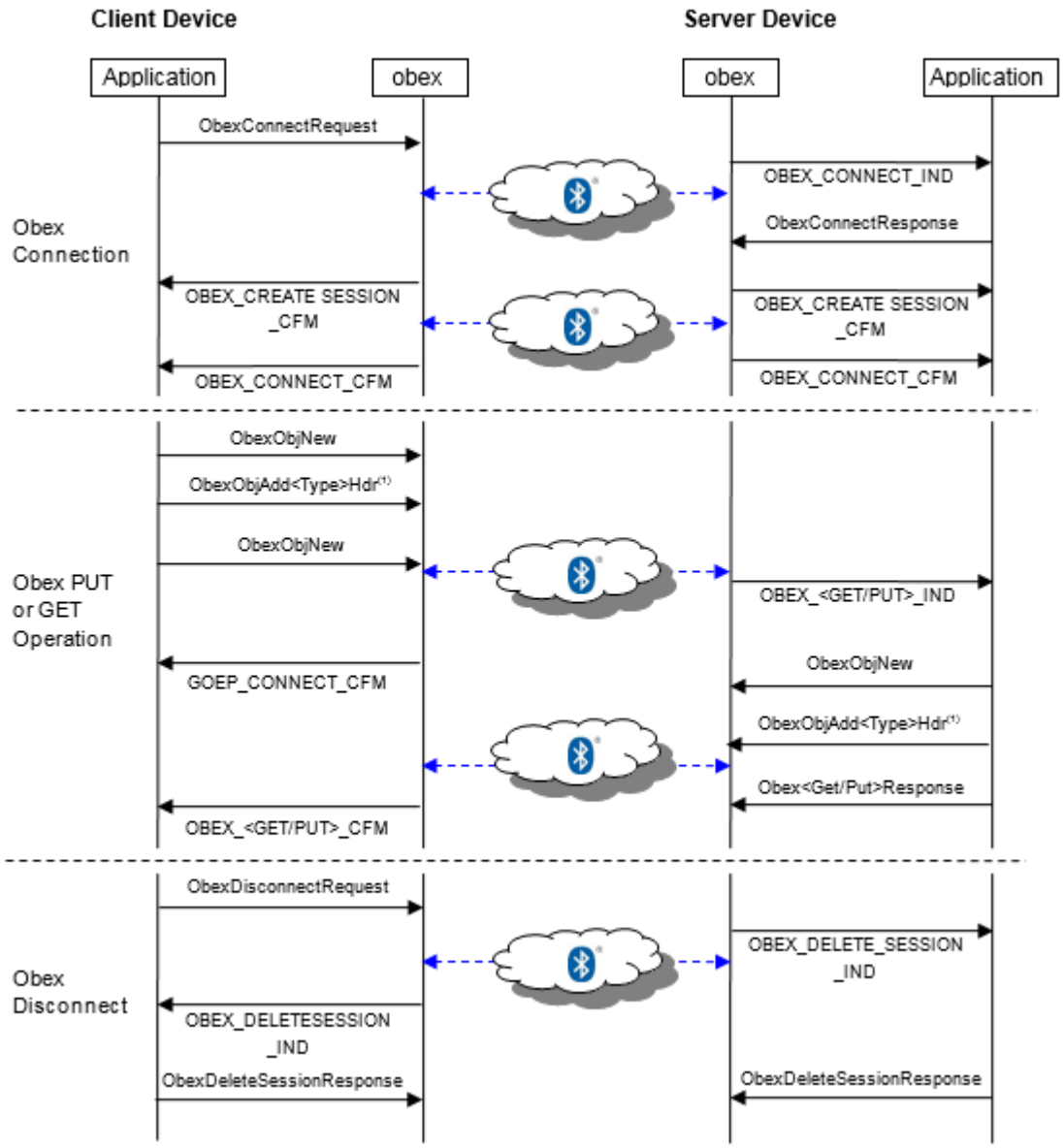
4.16 obex library

Interface to the Object Exchange Profile library.

Purpose

This library facilitates the base functionality on which the OPP, MAP, PBAP and FTP Profiles depend.

Typical message sequence chart



NOTE (1) Add one or more headers.

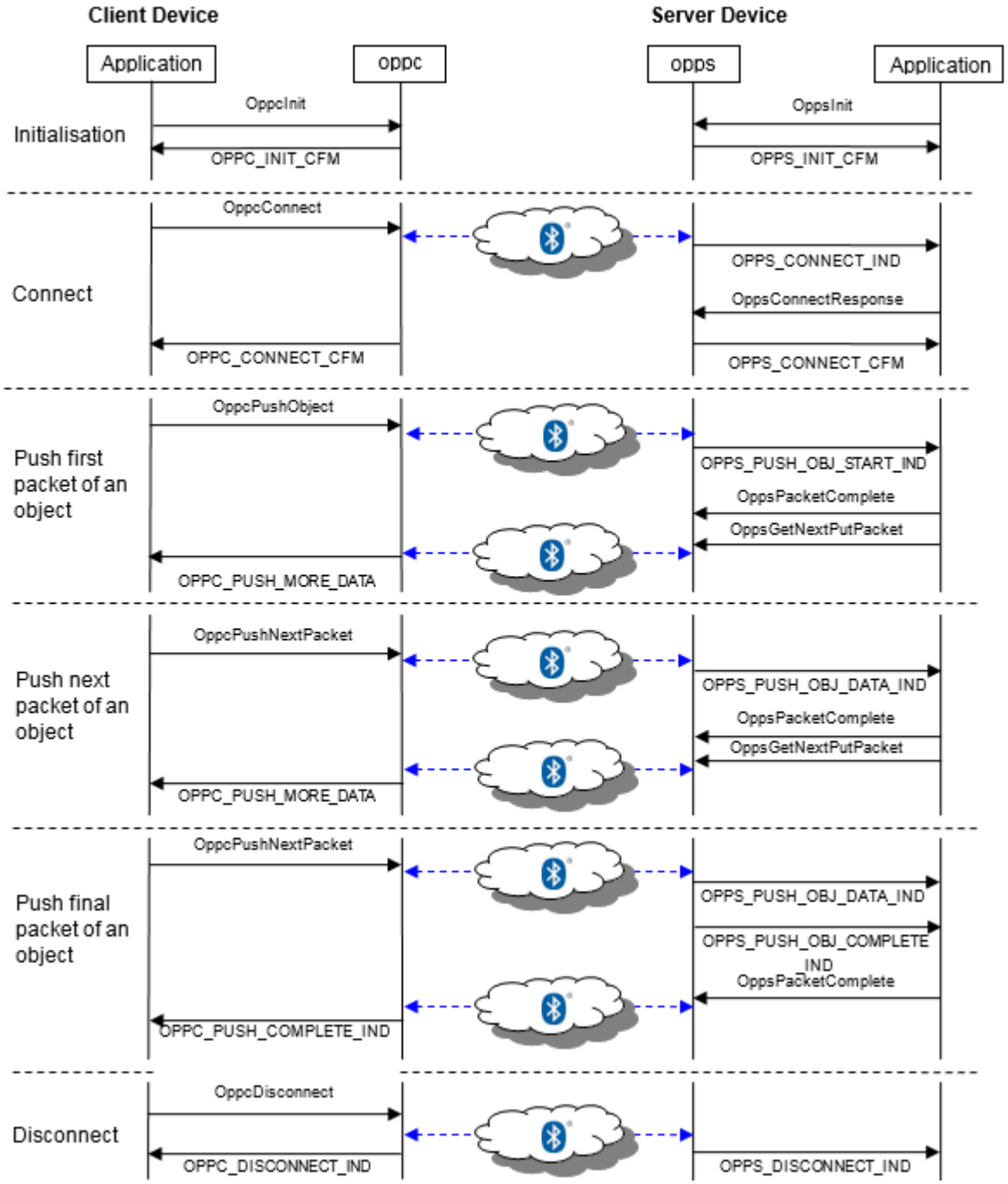
4.17 oppc library

Interface to the Object Push Protocol (OPP) Client Profile library.

Purpose

This library facilitates Object Push Client functionality, enabling objects to be pushed to and pulled from a remote device that is acting as an OPP server.

Typical message sequence chart



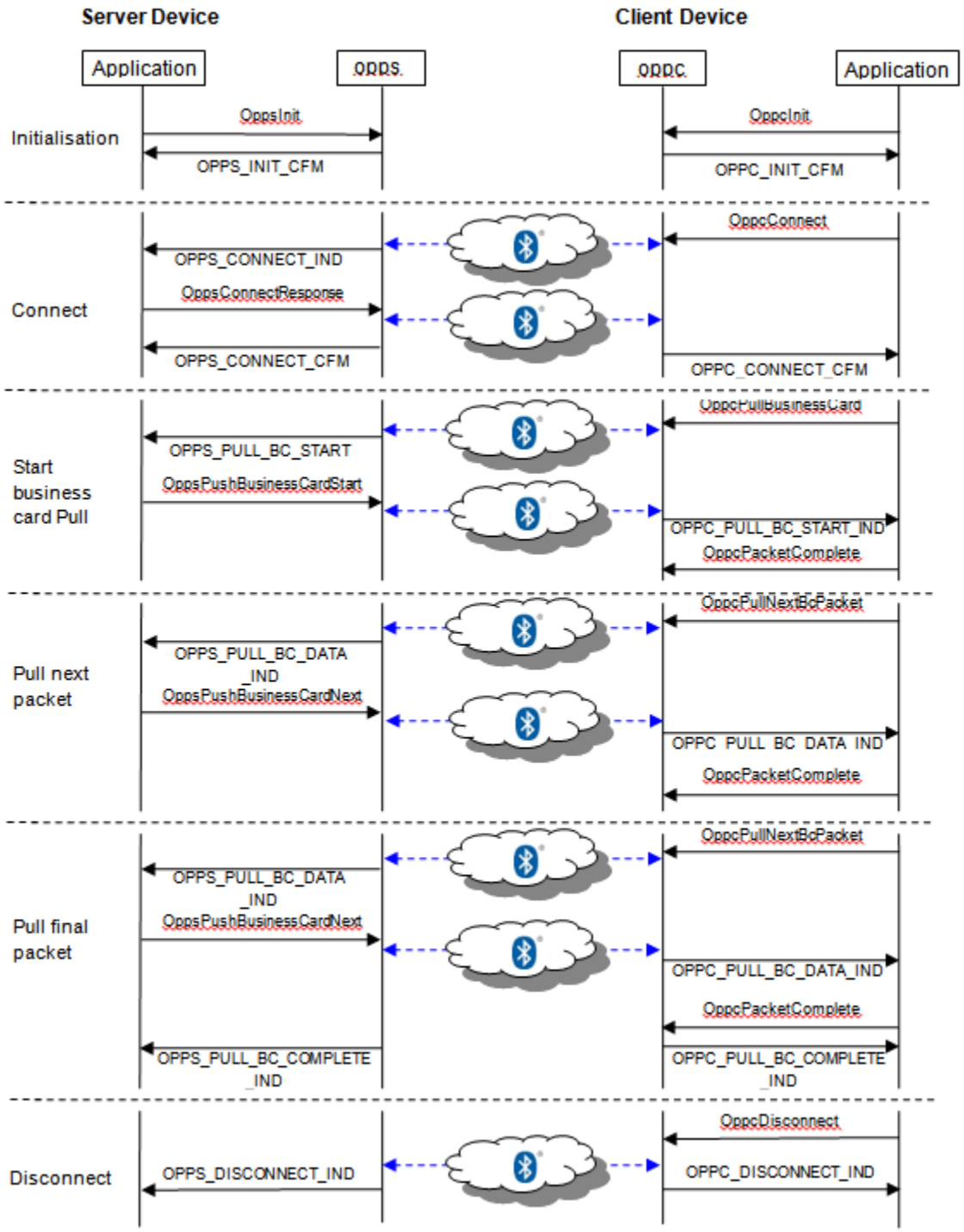
4.18 opps library

Interface to the OPP Server Profile library.

Purpose

This library facilitates Object Push Server functionality, enabling objects to be pushed to and pulled from the server under the control of an OPP client.

Typical message sequence chart



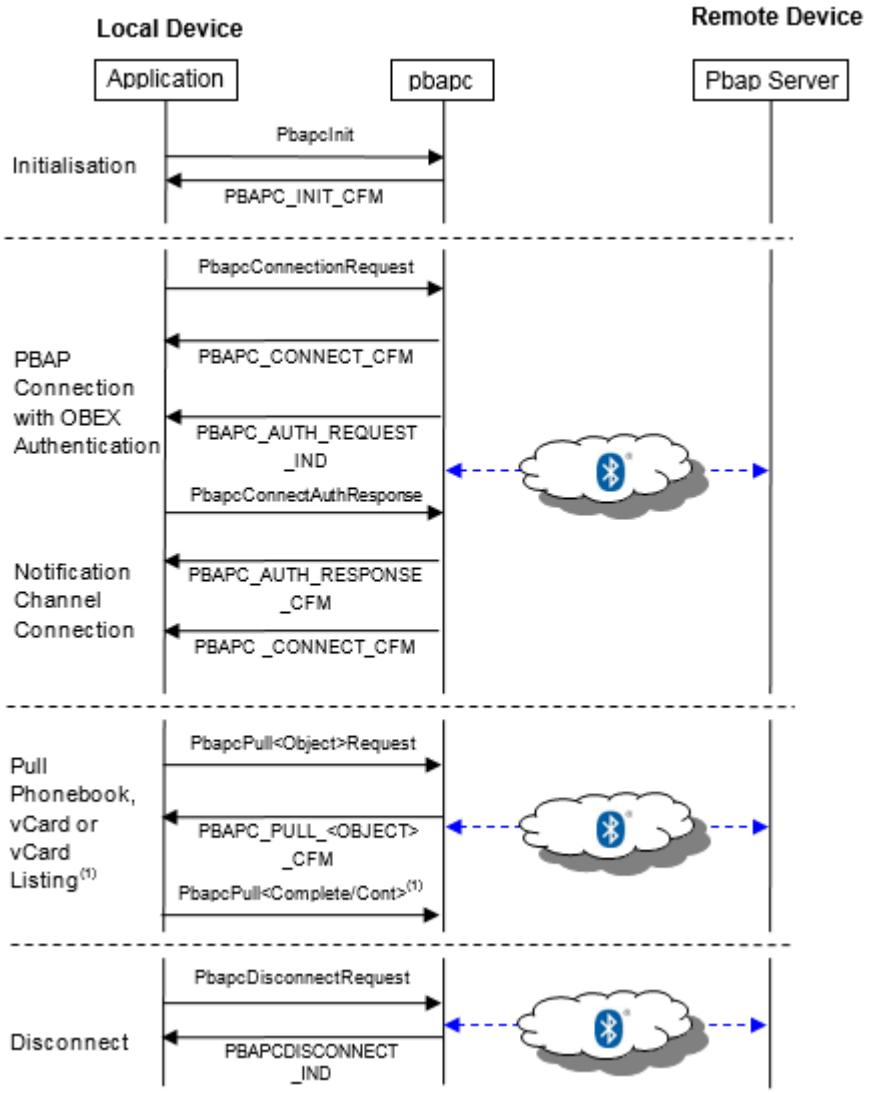
4.19 pbapc library

Interface to the Phonebook Access Profile library.

Purpose

This library facilitates exchange of objects such as contacts, call and calendar information, and so on, held as phonebook entries. It enables objects to be pushed to and pulled from the server under the control of a PBAP client.

Typical message sequence charts



NOTE (1) If the `PBAPC_PULL_<OBJECT>_CFM` returns with a status of Pending send `PbapcPull<Continue>`.
If it returns a status of Success send `PbapcPull<Complete>`. See the VM and Native Reference Guide in xIDE for descriptions of `pbapc` functions.

4.20 piolib library

This library allows more than one task to receive MESSAGE_PIO_CHANGED messages.

Purpose

This library can be used when an application has two or more tasks that are interested in PIO changes and need to receive the same PIO_CHANGED messages.

General

To use the library functions the library is first initialized by calling. PiolibInit and the PIO sub system is setup by calling PioDebounce.

Individual tasks interested in PIO messages can then call PioibRegister. Registered task(s) will receive PIO messages, until the task deregisters by calling PioDeregister.

This library cannot co-exist with the code auto-generated by buttonparse.exe.

4.21 power library

Interface to the Battery Reporter service library.

Purpose

This library allows readings of the test pins and supply voltage to be taken, either once or at regular intervals. The readings are sent as messages to the task supplied when initializing the battery library.

4.22 print library

Interface to the print library.

Purpose

This library allows applications to contain debug printfs in their code and the facility to enable/disable them as required, using a single switch:

Example

Debug printfs can be added to code:

```
#include <print.h>
PRINT (( "debug message\n" ))
```

The debug printfs can then be enabled or disabled:

- Enable printfs by adding DEBUG_PRINT_ENABLED to the Define symbols field of the application project properties and building the application in xIDE.
- Disable printfs by removing DEBUG_PRINT_ENABLED and rebuilding the application.

When using the facilities provided by print.h it is not necessary to add print to the list of libraries in the xIDE **Project Properties**.

4.23 region library

Interface to the region library.

Purpose

This library is used to write and read values to regions of uint8 memory and to check that the contents of a region matches a specific value.

This library is usually used in conjunction with the service library, which locates the regions inside a Bluetooth service record.

Example of use

Functions from the region library used in conjunction with functions from the service library would be used to retrieve, for example, the RFCOMM server channel from a service record.

4.24 service library

This is a helper library that facilitates the search of service records.

Purpose

This library provides functions that can be used to access regions of interest within service records. The functions return True if the record is found and False if the record is not found or is malformed.

Example

Examples of the use of the `service.h` library is most easily demonstrated by examining their use in source code, please see the following files:

- `<SDK Installation Folder>\src\lib\spp\spp_sdp.c`
- `<SDK Installation Folder>\src\lib\hfp\hfp_sdp.c`

4.25 sppc, spps and spp_common libraries

Interface to the Serial Port Profile (SPP) library.

The `sppc.h` file is for Serial Port Profile client applications and `spps.h` file is for Serial Port Profile server applications. The `spp_common.h` file is common to both.

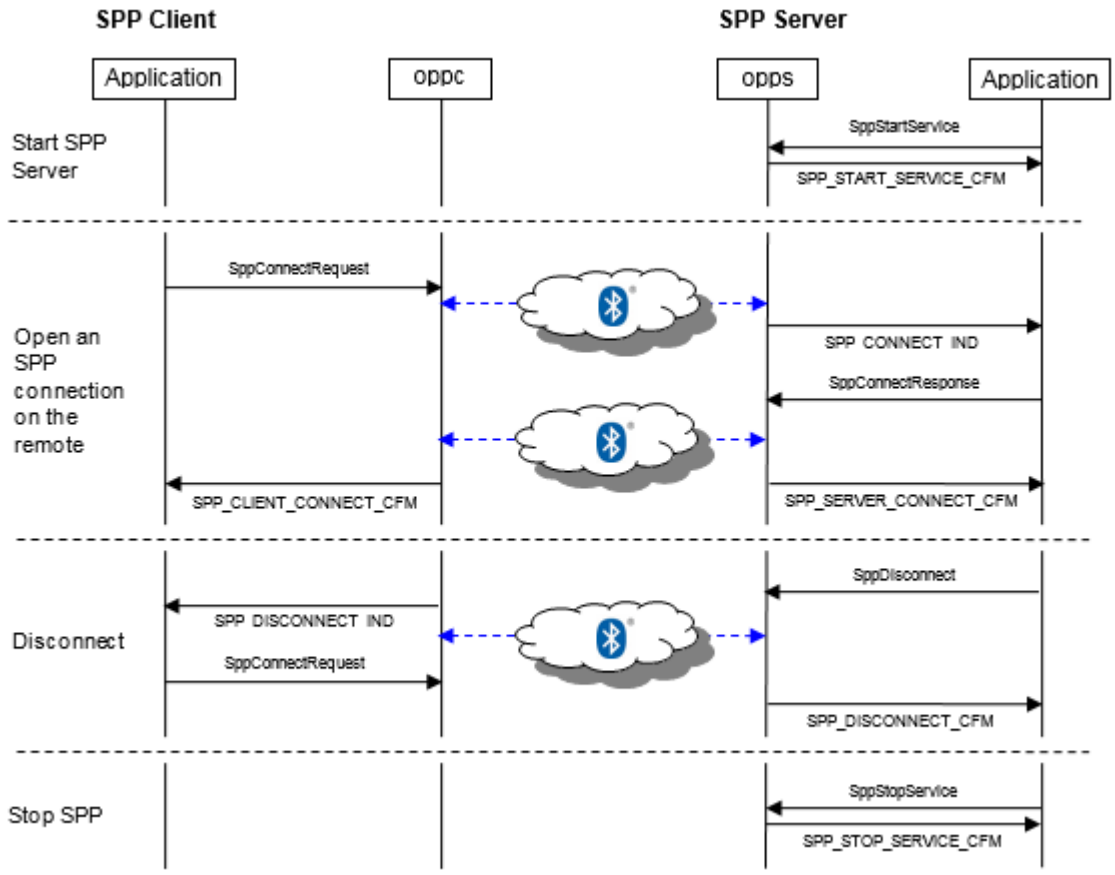
Purpose

These libraries implement the Serial Port Profile Specification. The SPP is used to set up an emulated serial cable connection using RFCOMM between two peer devices. The RFCOMM session exists on an L2CAP channel.

The `spps` library handles the registration of a service record with the SDP database, so that the service/applications can be reached using RFCOMM.

These libraries exposes a functional downstream API and an upstream message-based API.

Typical message sequence chart



5 Technical support

Further information on all QTIL products can be found on the (createpoint) website.

Document references

Document	Reference
A2DP Library User Guide	80-CT401-1/CS-00116660-UG
Connection Library RFCOMM API	80-CT434-1/CS-00207134-AN
HFP Library User Guide	80-CT427-1/CS-00206909-UG
Writing Qualcomm BlueCore Applications	80-CT402-1/CS-00207482-UG

Terms and definitions

Term	Definition
A2DP	Advanced Audio Distribution Profile
ADC	Analogue to Digital Converter
ADK	Audio or Application Development Kit
AGHFP	Audio Gateway Hands Free Profile
API	Application Programming Interface
BLE	Bluetooth Low Energy
BlueCore	Group term for the QTIL range of Bluetooth wireless technology ICs
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
BR	Basic Rate
CaSiRa	QTIL Bluetooth development hardware
CT	Controller device
DAC	Digital to Analogue Converter
DM	Device Manager
DSP	Digital Signal Processor: a microprocessor dedicated to real-time signal processing.
DUN	Dial Up Networking
EDR	Enhanced Data Rate
FM	Frequency Modulation
FTP	File Transport Protocol
GATT	Generic Attribute Profile
GAVDP	Generic Audio Visual Distribution Profile
HCI	Host Controller Interface
HDP	Health Device Profile
HFP	Hands Free Profile
HID	Humuan Interface Device Profile
IC	Integrated Circuit
L2CAP	Logical Link Control and Adaptation Protocol
MAP	Message Access Profile
MAS	Message Access Service
MCE	Message Client Equipment

Term	Definition
MD5	Message Digest algorithm 5
MMI	Man Machine Interface
MNS	Message Notification Service
MSE	Message Server Equipment
OBEX	Object EXchange
OPP	Object Push Profile
PBAP	Phonebook Access Profile
PSM	Protocol/Service Multiplexer
RFCOMM	Serial Cable Emulation Protocol based on ETSI TS 07.10.
Rx	Receive
SCO	Synchronous Connection Orientation link
SDK	Software Development Kit
SDP	Service Discovery Protocol
SEP	Stream End Point
SIG	(Bluetooth) Special Interest Group
SNK	(Data) Sink
SPP	Serial Port Profile
SRC	(Data) Source
TG	Target device
VM	Virtual Machine
xIDE	The QTIL Bluetooth Integrated Development Environment