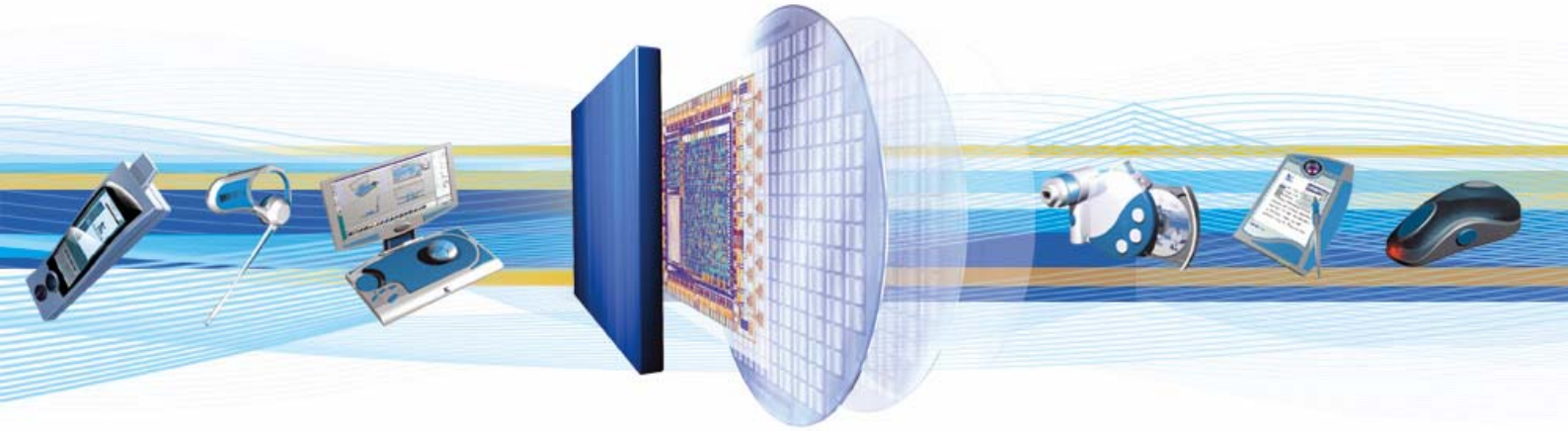
a guide to

BlueLab™ Libraries

May 2005



CSR

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

Registered in England 3665875

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

Contents

1 Introduction

This document describes the libraries provided as part of BlueLab™.

1.1 General

The BlueLab libraries sit between the application and the BlueCore firmware.

For software engineers to successfully develop Bluetooth applications using BlueLab, it is essential to gain an understanding and working knowledge of the BlueLab libraries.

It is not necessary to know all the function calls in every library. This information is available in the 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 will also help software engineers new to Bluetooth to interpret the reference application code, which in turn indicate the correct way to use the libraries.

A knowledge of the libraries can also reduce the development effort by avoiding unnecessary rewriting of code for functionality already implemented within the libraries.

2 Overview of Libraries

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 CSR's BlueCore chips.

2.1 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 BlueLab.

The Libraries that fall into this category are:

Standard C 'Foundation' Libraries	
assert	Allows assertions to be activated/deactivated using the NDEBUG macro identifier, to aid documentation and debugging of code
ctype	Implements character handling functions in a non system dependent way
iso646	Defines macros for alternative representation of logical operators
limits	Defines the maximum/minimum limits of basic data types
memory	Implements certain dynamic memory allocation/deallocation
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 undefined 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: BlueLab 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.	

CSR 'Foundation' Libraries	
adc	Functions used to obtain voltage readings from the Analogue to Digital Converter (ADC) hardware.
audio_notes	Macros used to generate tones
boot	Used to control BlueCore boot mode on unified firmware
energy	Functions used to enable and disable energy estimation on a given SCO connection
file	Functions used to access the read-only file-system
host	Function used to send a message to the host
i2c	Allows transfer of data across the I^2C interface
kalimba	Functions used to control the Digital Signal Processor (DSP) on BlueCore Multimedia chips
link	Enables link key snooping for dual boot devices
message	Functions used to control message passing
panic	Functions that can be used to panic the application
pcm	Functions used to control the Pulse Code Modulation (PCM) hardware
pio	Functions used to access BlueCore Input/Output lines
ps	Provides access to the persistent key store on Bluecore
sink	Functions used to output data on 8-bit streams
source	Functions used to handle data arriving on 8-bit streams
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
vgen	Interface definition for vThing (vCard, vCal etc) file generation
vm	Functions providing low level access to BlueCore firmware and hardware

2.1.1 Using Foundation Libraries

Both the standard C and CSR 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:

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

  Note: This saves the developer having to specify individual libraries but will increase the time taken to compile the code.

- **csrtypes:** this library declares all the CSR specific types used in the support and application libraries. Any other header file that requires the typedefs in the csrtypes library itself includes the `csrtypes.h`.

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

2.2 Support Libraries

These comprise a set of libraries that help the connection layer 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 eg battery status readings.

The Libraries that fall into this category are:

battery	kalimba_standard_messages
bdaddr	print
codec	region
connection	service

The header files (`.h` files) and where relevant the source code (`.c` files) for these libraries can be found in `C:\BlueLab\src\lib`.

To use a support library its header file must be `#included` in the application source file and where appropriate the library must be listed in the xIDE Project Properties, see Chapter 4.

2.3 Profile Libraries

These libraries implement the Bluetooth Profiles as defined in the Bluetooth specification.

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	goep
avrcp	hfp
ftpc	oppc
ftps	opps
gavdp	spp

The header files (`.h` files) and source code (`.c` files) for these libraries can be found in `C:\BlueLab\src\lib`

To use a profile library its header file must be `#included` in the application source file and the library must be listed in the xIDE Project Properties, see Chapter 4.

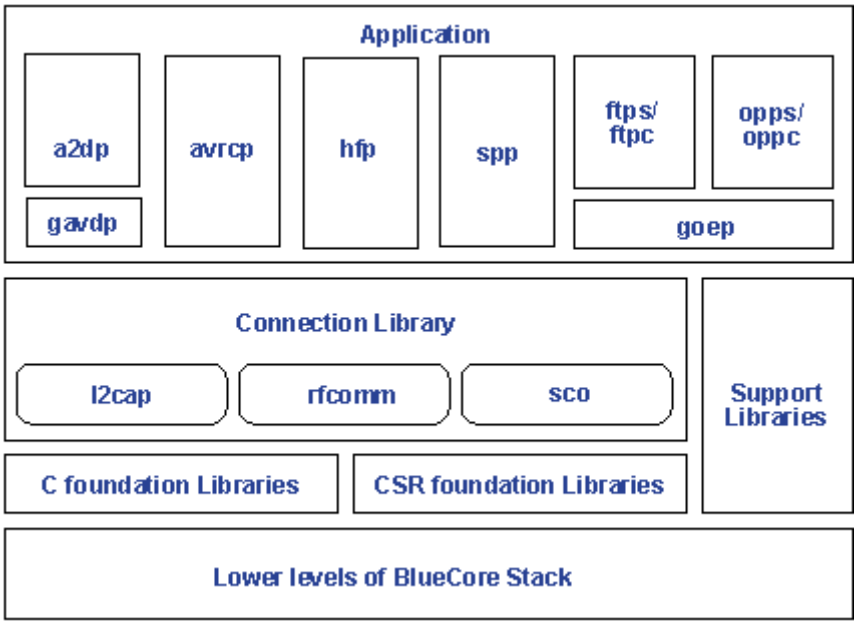
Note: The properties of supplied projects, ie example and references application source code, will automatically list the libraries used by the code.

.

3 Library Architecture and dependency

The library functions supplied with BlueLab expose the BlueCore stack to the application code in structured layers.

The diagram below gives a conceptual representation of the library hierarchy:



3.1 Library interaction

In order to better understand how the libraries interact, it is useful to consider one of the reference applications provided in BlueLab and how functions from other libraries can be used to extend functionality.

The `av_headset` reference application uses functions from the libraries shown in table below.

Libraries incorporated in the av_headset reference application	
a2dp	This library implements the a2dp using the services of the gavdp library. It permits an audio stream to be configured, started, stopped and suspended. The actual streaming of data is performed by the underlying firmware.
gavdp	This library defines the binary transactions between Bluetooth devices for stream setup and media streaming. This library is accessed through the a2dp layer and is not directly exposed to the application.
connection	This library handles gavdp messages, creating and controlling the l2CAP connections.
bdaddr	This support library facilitates the comparison of Bluetooth addresses when negotiating a connection between Bluetooth devices
Note: For clarity this table does not consider the library functions implementing the DSP functionality required by the av_headset reference application.	

3.1.1 Adding functionality using BlueLab libraries

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

In most cases it is expected that further functionality will be required in order to provide a commercial application.

For example the introduction of code to monitor the battery status would probably be required in most commercial applications.

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 eg function types, parameters passed etc can be found in the Reference Guide, which is part of the xIDE on-line Help documentation.

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

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

For example, to add hands-free functionality to the `av_headset` reference application the `hfp` library is used. Because the `hfp` library uses functions in the `service` and `region` libraries all three would need to be added. This can be seen in the `av_headset_hfp` reference application supplied with BlueLab.

Comparing the `av_headset` and `av_headset_hfp` reference applications shows how this can be implemented in practice.

4 Using BlueLab 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 `#included` 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 `library.a` file.

4.1 Including 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. eg

```
#include "header.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.

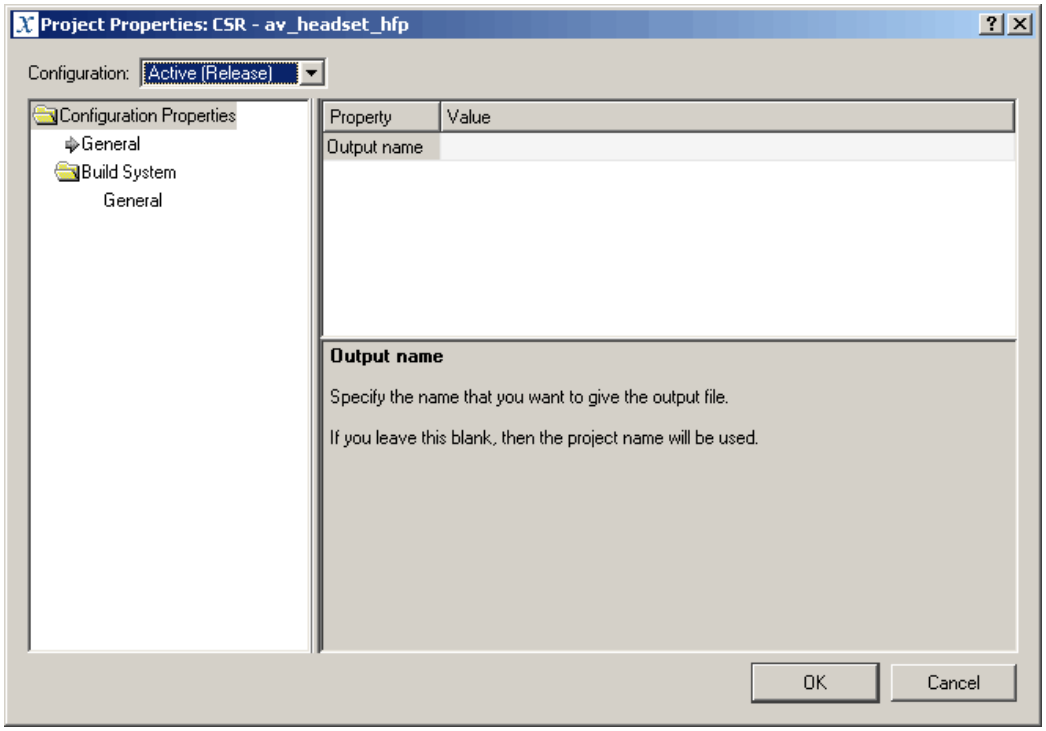
4.2 Adding a library to Project Properties

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:

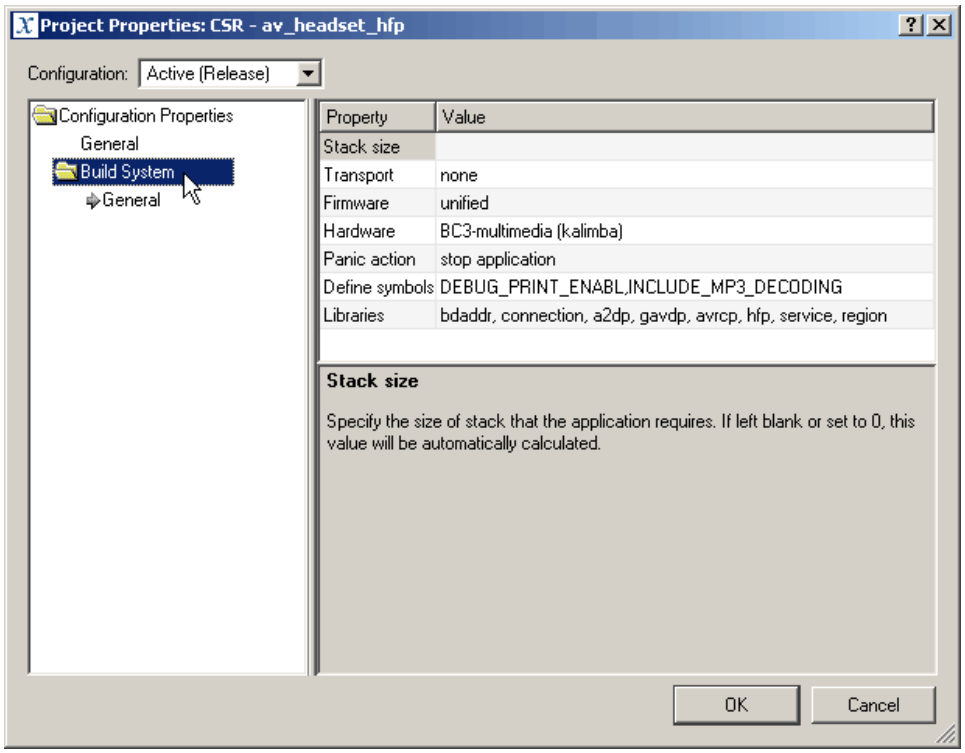
1. Select **Properties** from the xIDE **Project** menu.

   The Project Properties dialog appears:



2. Click on the **Build System** folder.

   The Build System properties are displayed in the right-hand workspace:



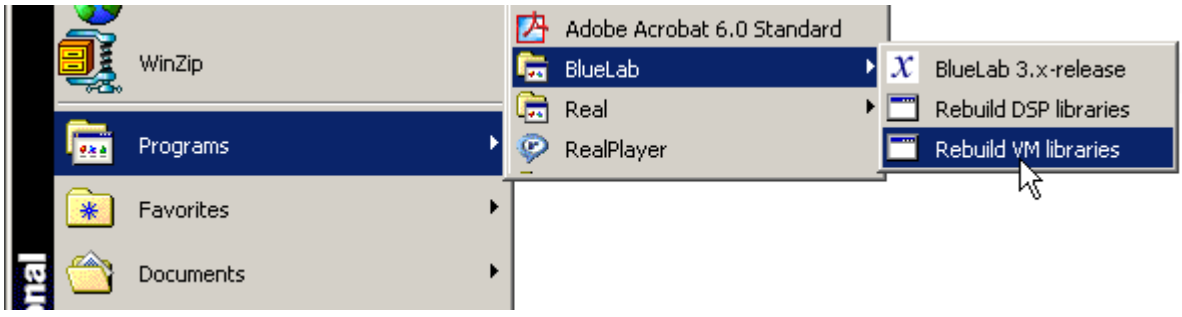
3. Click on the **Libraries** row to activate the **Value** field.
4. When the required libraries have been added, click **OK** to set the properties for the project.

4.3 Building / rebuilding Libraries

The BlueLab libraries are built as part of the BlueLab 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/BlueLab** and select the libraries you want to rebuild:



5 Library Reference

This chapter gives an overview of the functionality provided by the BlueLab Profile and Support libraries.

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

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

5.1 General

The naming conventions employed in BlueLab 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.

> eg the function `A2dpOpen` opens a local Stream End Point (SEP) and causes the confirmation message `A2DP_OPEN_CFM_T` to be returned.
>
> When a remote device opens a local SEP the application receives the following indication message `A2DP_OPEN_IND_T`.

When writing an application using BlueLab, CSR recommend that the primary consideration is the profile(s) that is/are to be implemented, rather than any other requirements such as the type of connection.

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

Libraries described in this Chapter:

BlueLab Profile and Support Libraries	
a2dp	Interface to the Advanced Audio Distribution Profile library
avrcp	Interface to the Audio Video Remote Control Profile library
battery	Take battery readings and return them in millivolts
bdaddr	Helper routines for Bluetooth addresses
codec	This library implements the functionality required to configure an internal or external stereo codec for use.
connection	Used to facilitate L2CAP,RFCOMM and SCO connections
ftpc	Interface to the File Transfer Profile (FTP) Client library
ftps	Interface to the File Transfer Profile (FTP) Server library
gavdp	Interface to the Generic Audio Visual Distribution Profile (GAVDP) library
goep	Interface to the Generic Object Exchange Profile (GOEP) library
hfp	Interface to the Hands Free Profile (HFP) library
kalimba_standard_messages	The messages passed between the kalimba libraries and the VM application
oppc	Interface to the Object Push Profile (OPP) Client library
opps	Interface to the Object Push Profile (OPP) Server library
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

5.2 a2dp.h

Interface to the Advanced Audio Distribution Profile library.

5.2.1 Purpose

This profile library implements the A2DP using the services of the GAVDP library. (The GAVDP library is not expected to be used directly).

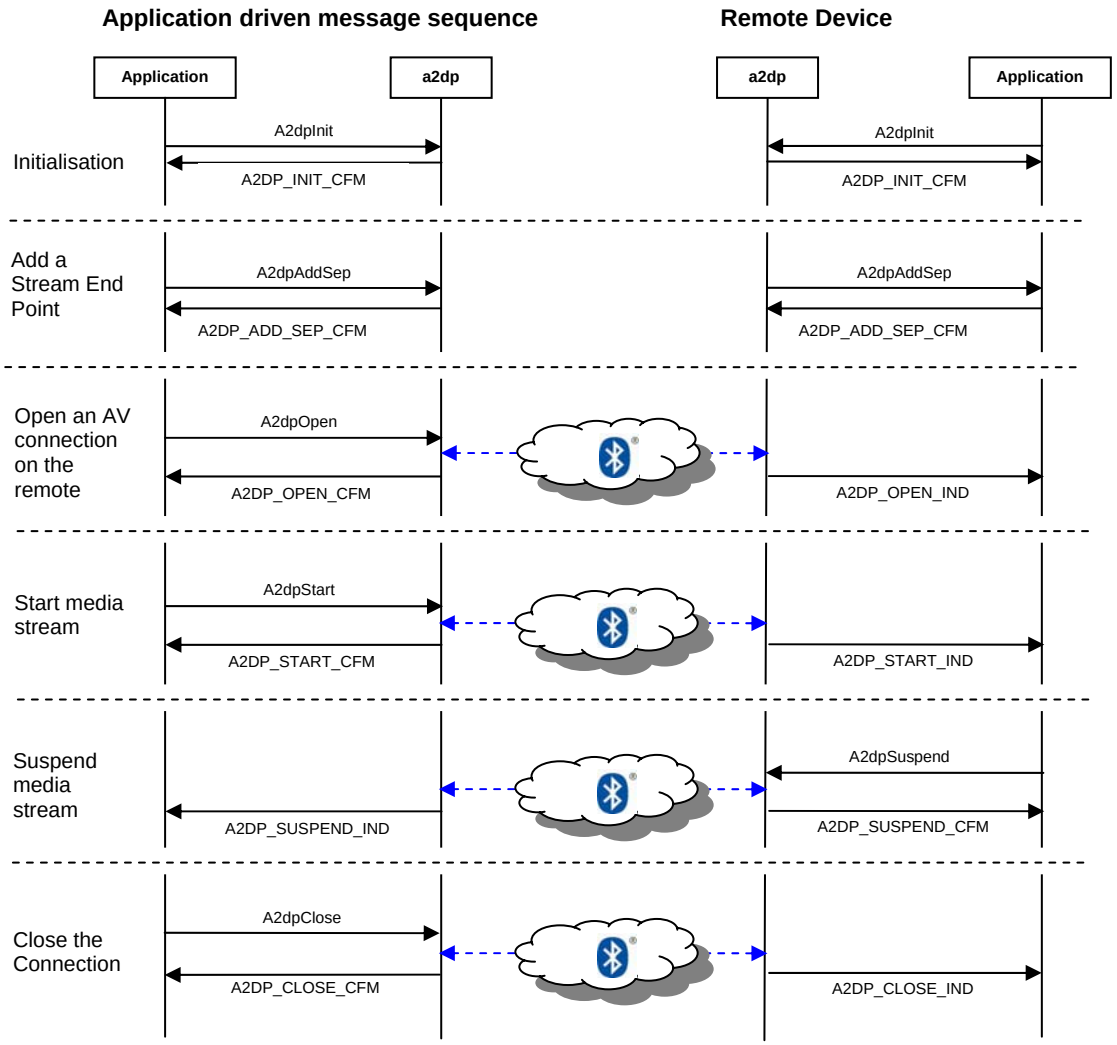
When a device wishes to start streaming audio content, the device must first set up a streaming connection. During the stream setup, the devices select the most suitable audio streaming parameters for the selected CODEC.

When a media transport channel is established and the start streaming procedure is executed, as defined in the GAVDP specification, audio can be streamed from the source (SRC) to the sink (SNK).

This library provides the low level services to permit an audio stream to be set up, configured, started and suspended. The audio stream is routed to the Digital Signal Processor (DSP) present on CSR BlueCore Multimedia devices. The CPU intensive operation of encoding/decoding a media stream is performed by the DSP. The data is then transferred over the Bluetooth link by the underlying firmware.

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

5.2.2 Typical Message Sequence Chart



5.3 avrcp.h

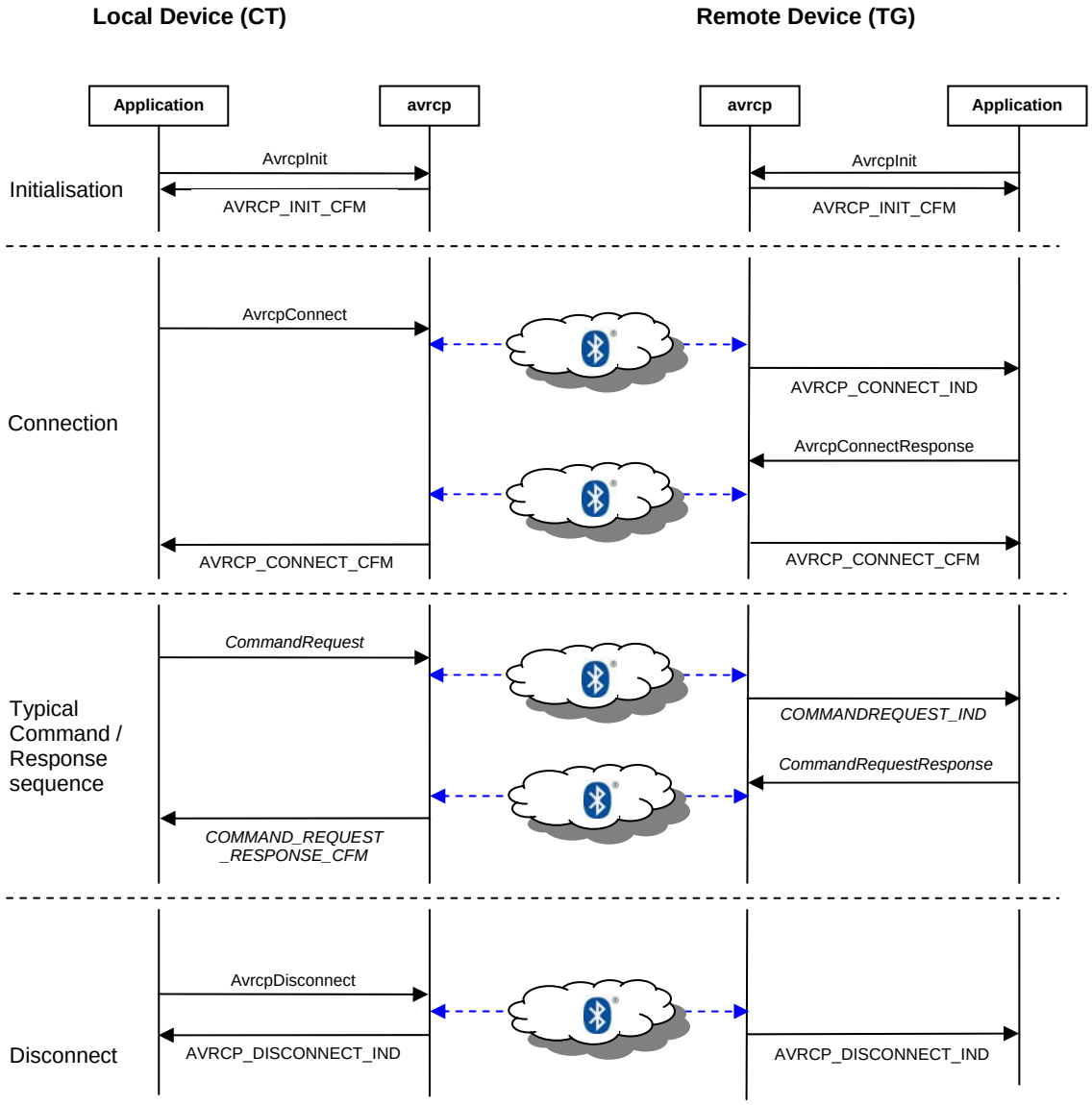
Interface to the Audio Video Remote Control Profile library.

5.3.1 Purpose

This Profile library implements the AVRCP. 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.

5.3.2 Typical Message Sequence Chart



Note: *Italic* labels in the sequence chart indicate generic functions and messages. See the 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).

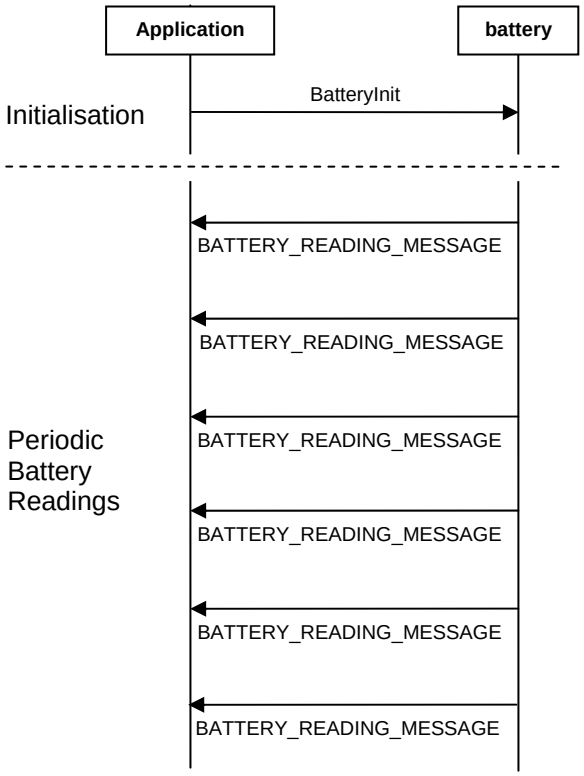
5.4 battery.h

Interface to allow battery monitoring and to return readings in millivolts.

5.4.1 Purpose

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 initialising the battery library.

5.4.2 Typical Message Sequence Chart



5.5 bdaddr.h

Interface to allow Bluetooth address handling.

5.5.1 Purpose

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

5.5.2 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.

5.6 codec.h

Interface to the CODEC library.

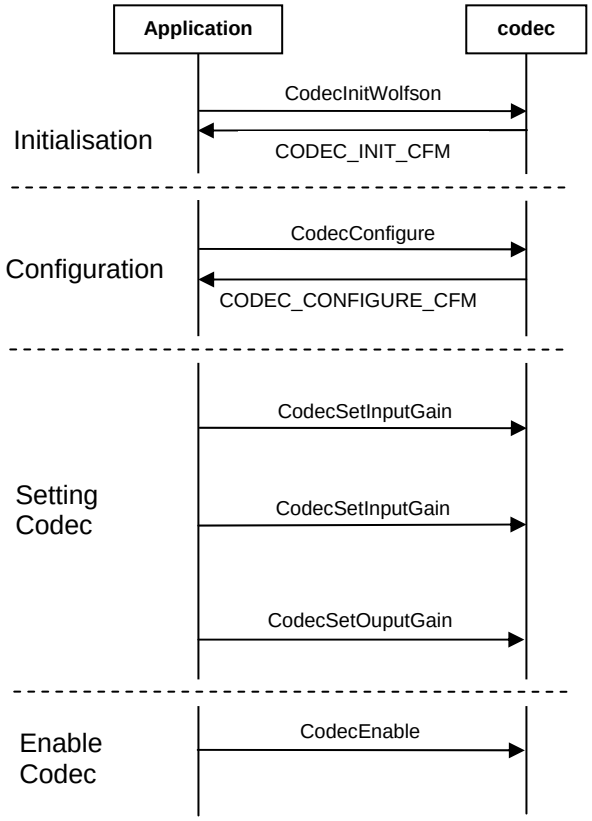
5.6.1 Purpose

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

It provides support for initialising, configuring and using a particular CODEC.

5.6.2 Typical Message Sequence Chart

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



5.7 connection.h

Interface to the connection library.

5.7.1 Purpose

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

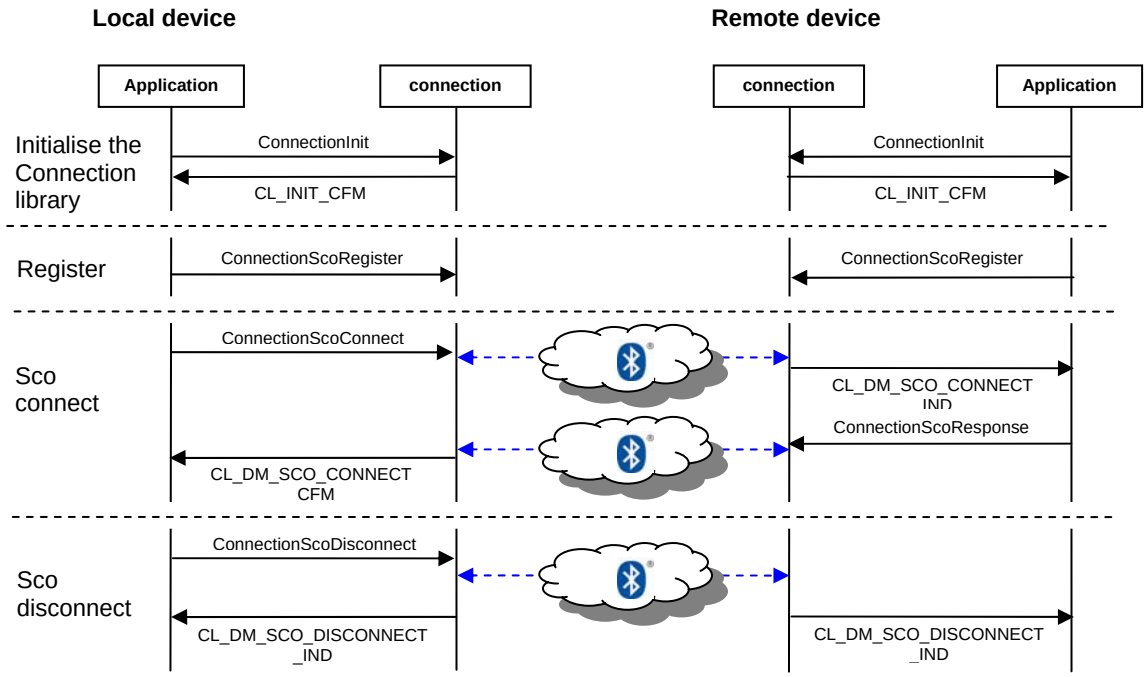
The services provided include support for RFCOMM, L2CAP and SCO connections, SDP search primitives and access to the Device Manager (DM) layer that exposes the Host Controller Interface (HCI) layer functionality.

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

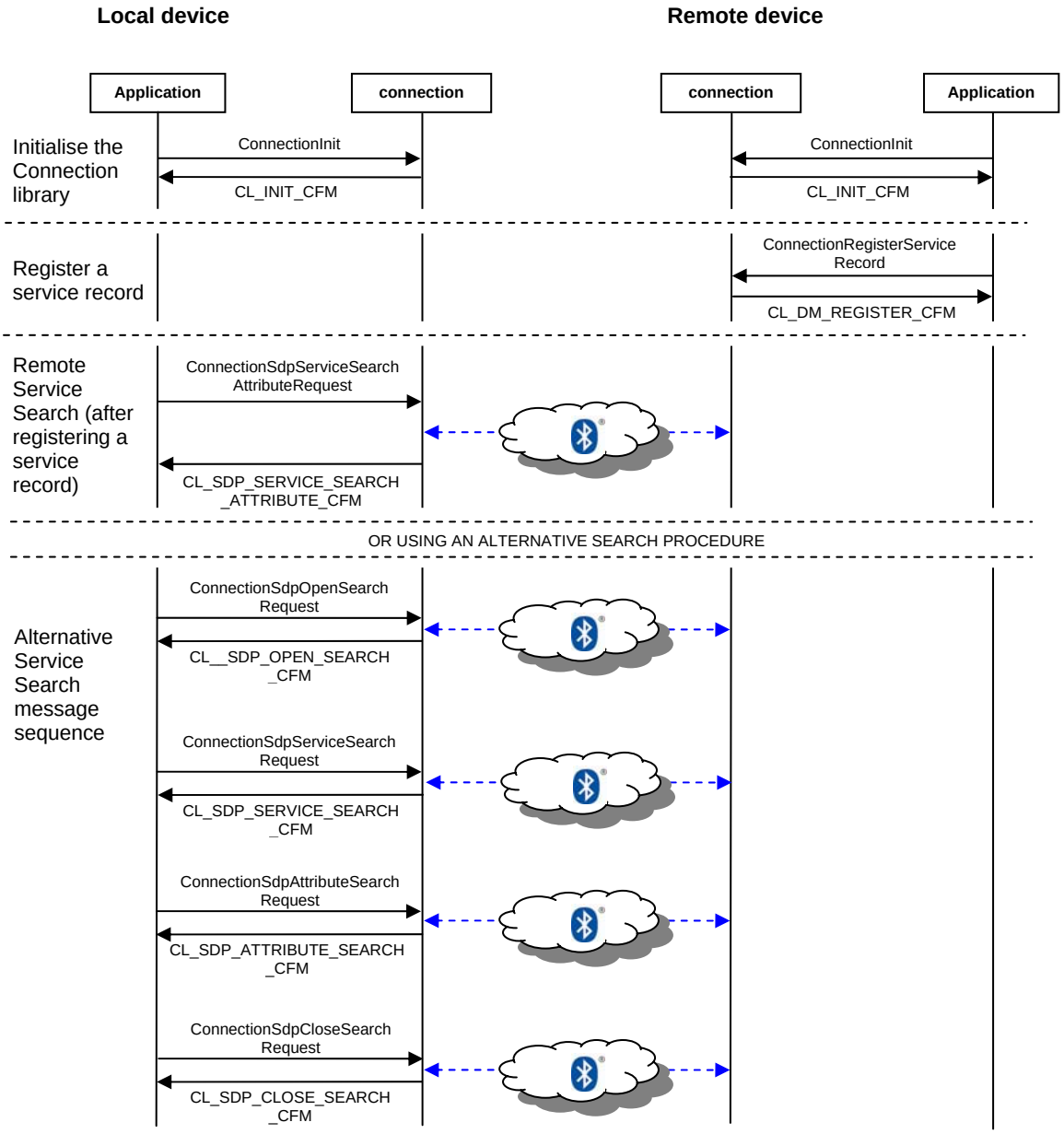
The xIDE on-line Reference Guide gives any additional information needed to use connection library functions in application code.

5.7.2 Typical Message Sequence Chart

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



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



Once a service record has been registered responses to service search requests are handled by Bluestack and are not seen by the Application.

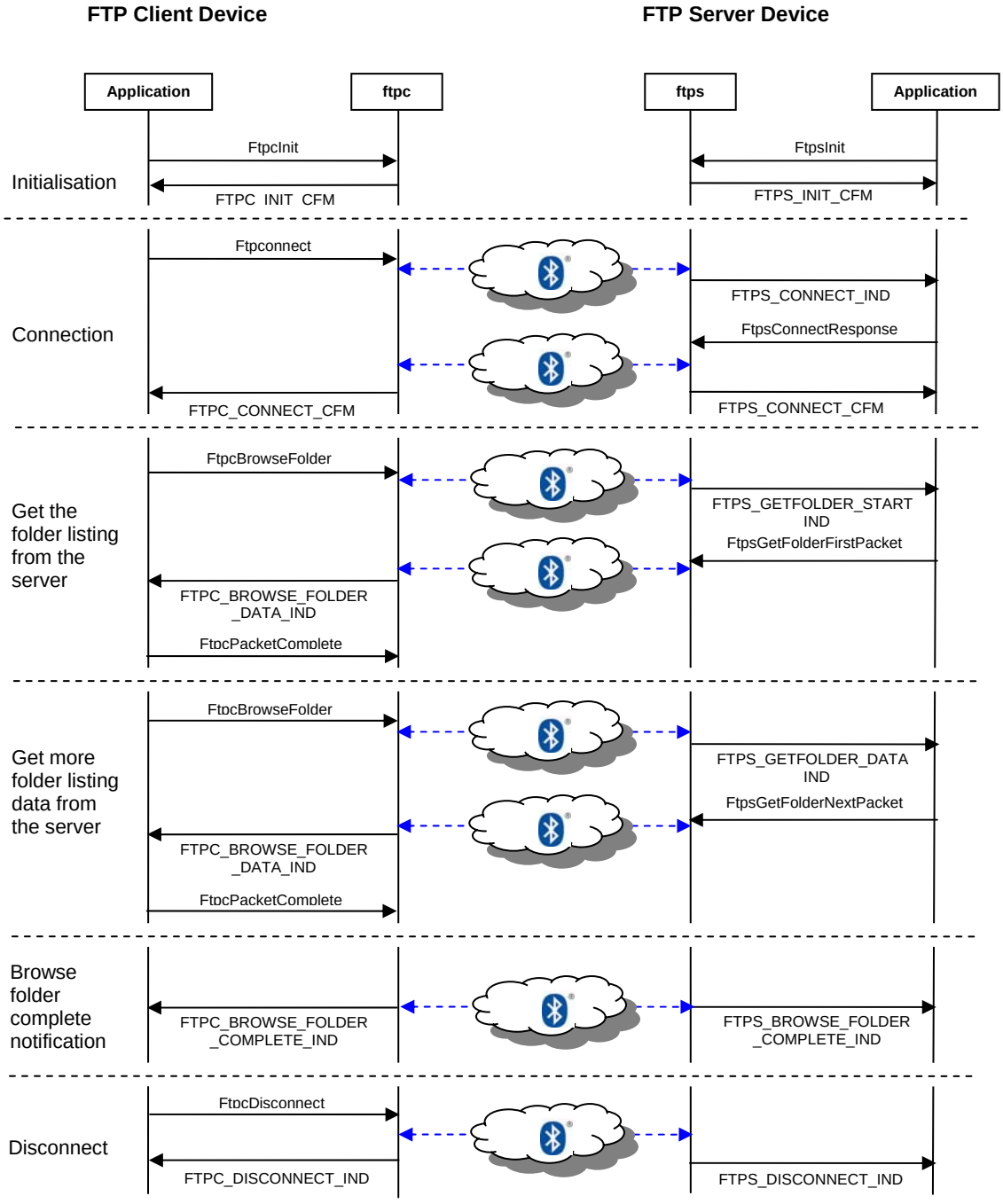
5.8 ftpc.h

Interface to the FTP Client Profile library.

5.8.1 Purpose

This library facilitates FTP Client functionality, enabling the initiation and control of an FTP session with a remote server.

5.8.2 Typical Message Sequence Chart



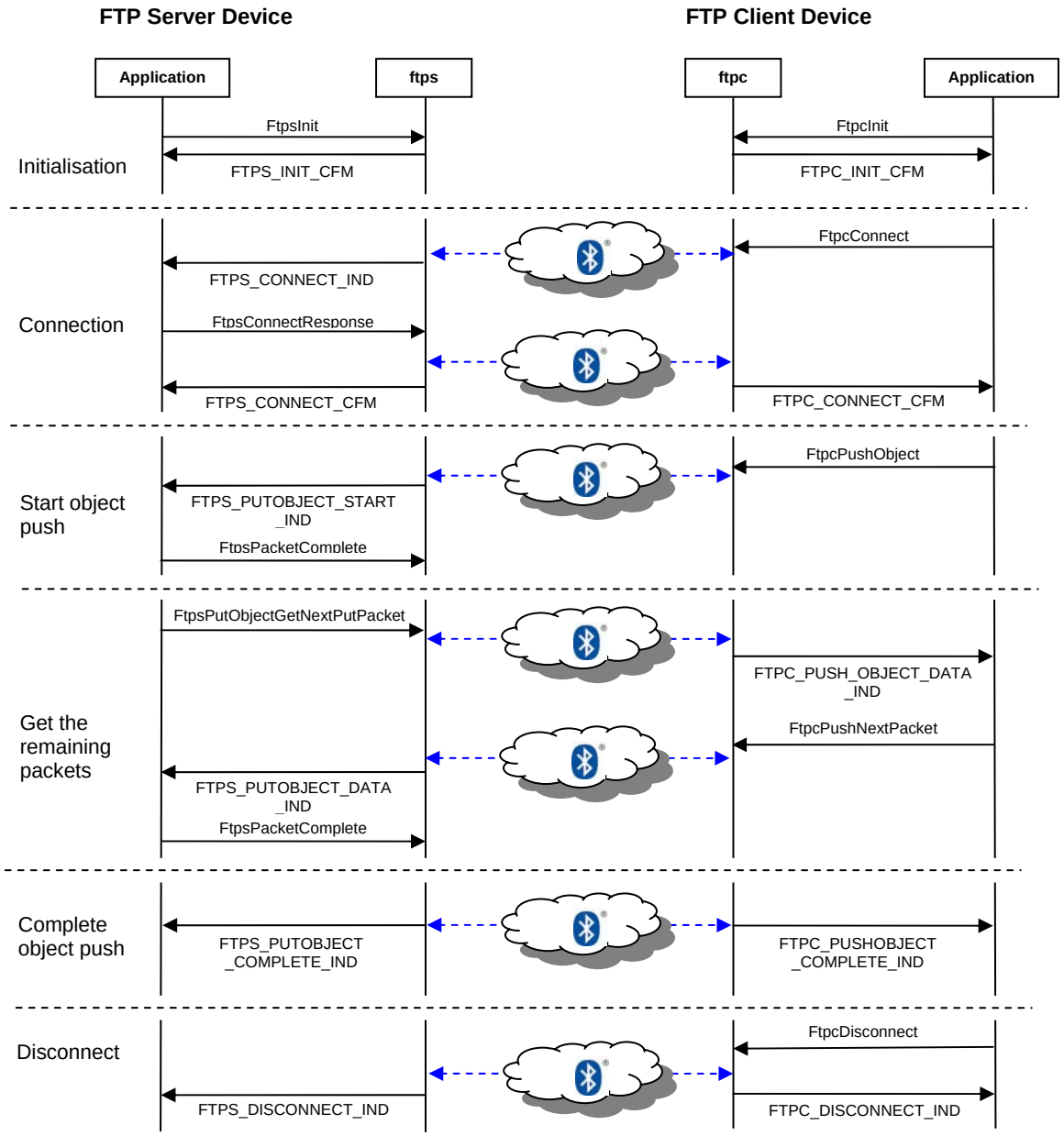
5.9 ftps.h

Interface to the FTP Server Profile library.

5.9.1 Purpose

This library facilitates FTP Server functionality, enabling the acceptance of FTP requests from a remote client.

5.9.2 Typical Message Sequence Chart



5.10 gavdp.h

Interface to the Generic Audio Visual Distribution Profile library.

5.10.1 Purpose

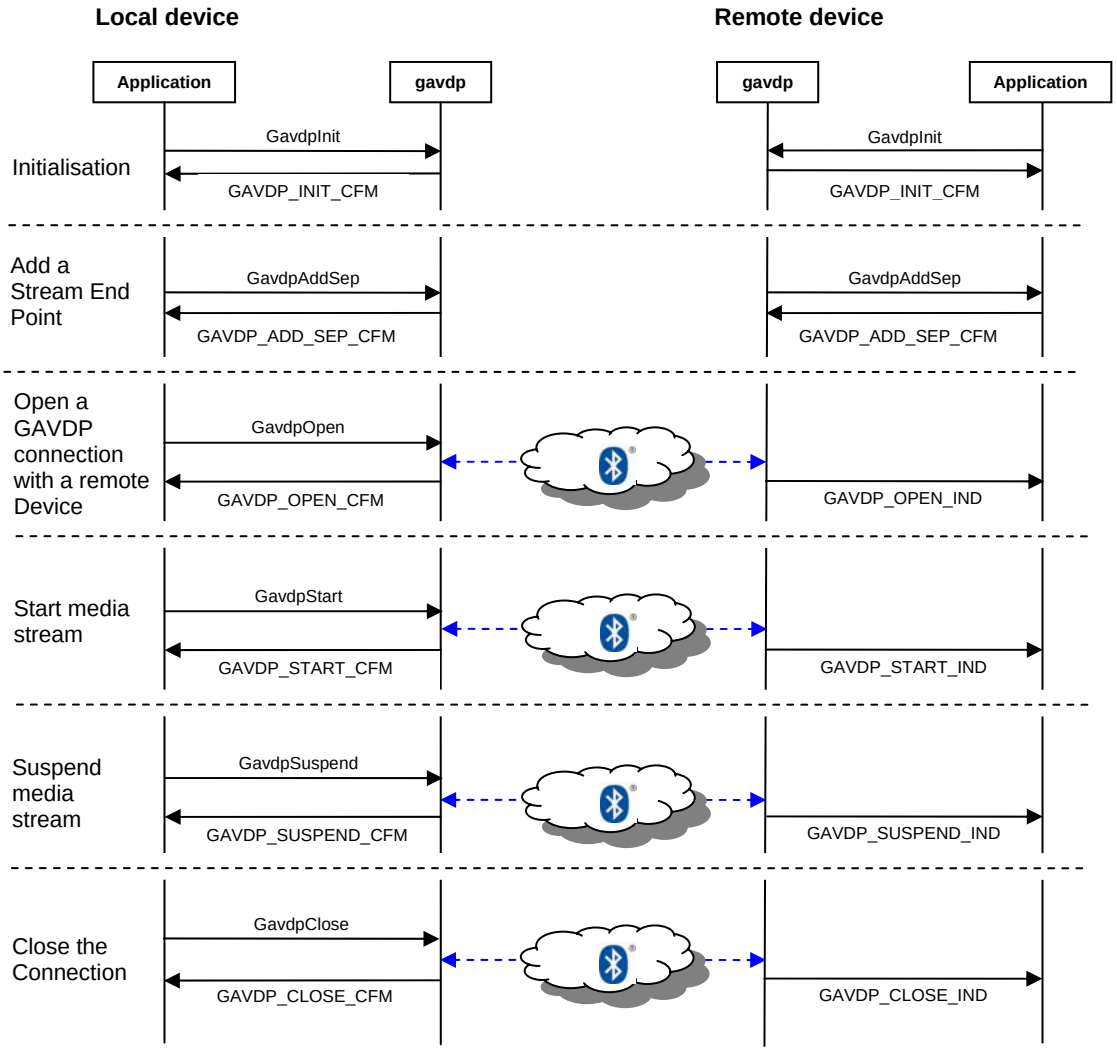
This library facilitates the set up and transport of Audio/Video streams over an L2CAP connection.

It implements the Audio/Visual Distribution Transport Protocol Specification.

A/V streaming and stream setup signalling are transported via L2CAP packets. A dedicated Protocol/Service Multiplexer (PSM) value is used to identify L2CAP packets that are intended for AVDTP.

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

5.10.2 Typical Message Sequence Chart



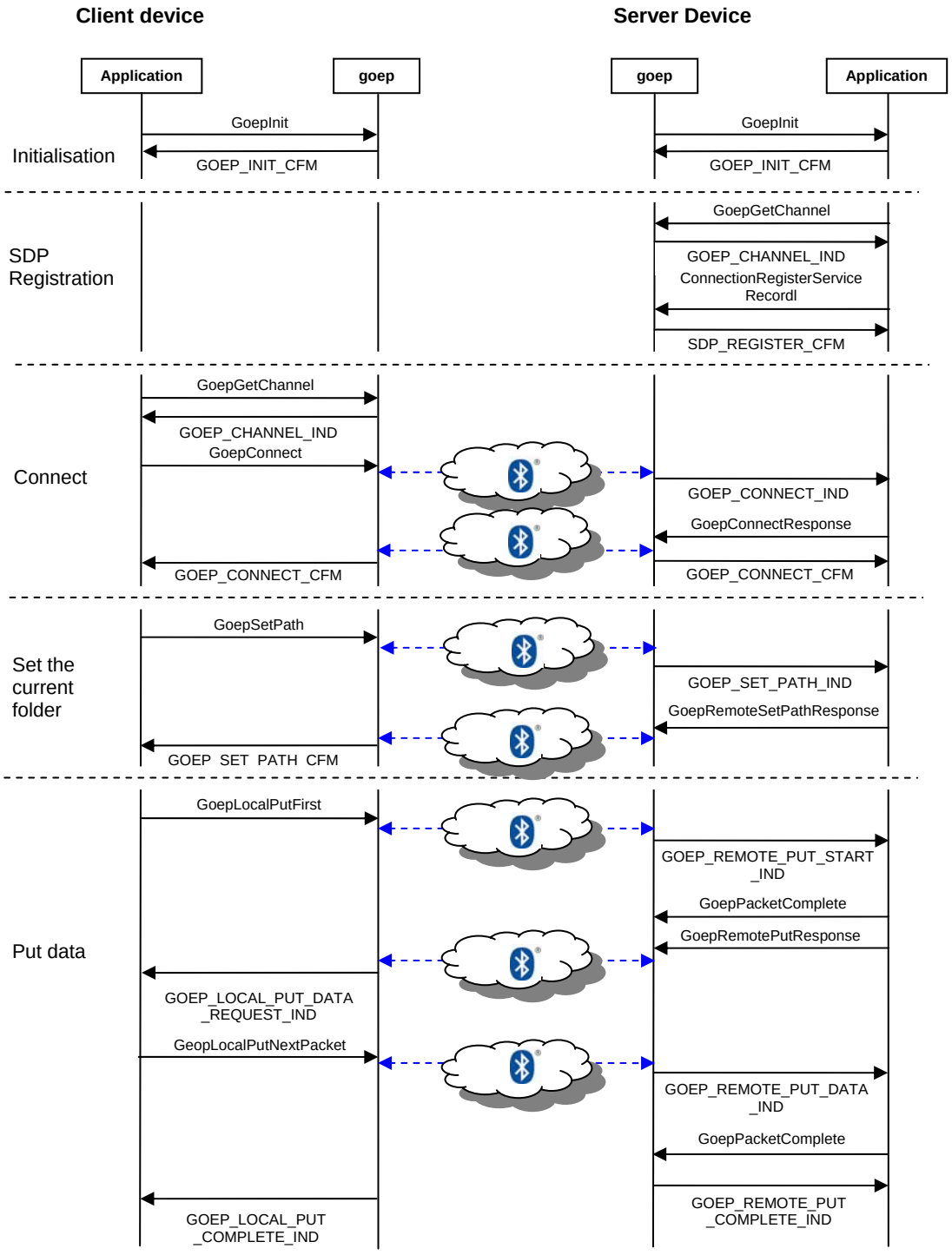
5.11 goep.h

Interface to the General Object Exchange Profile library.

5.11.1 Purpose

This library facilitates the base functionality on which the FTP Client, FTP Server, OPP Client and OPP Server Profiles depend.

5.11.2 Typical Message Sequence Chart



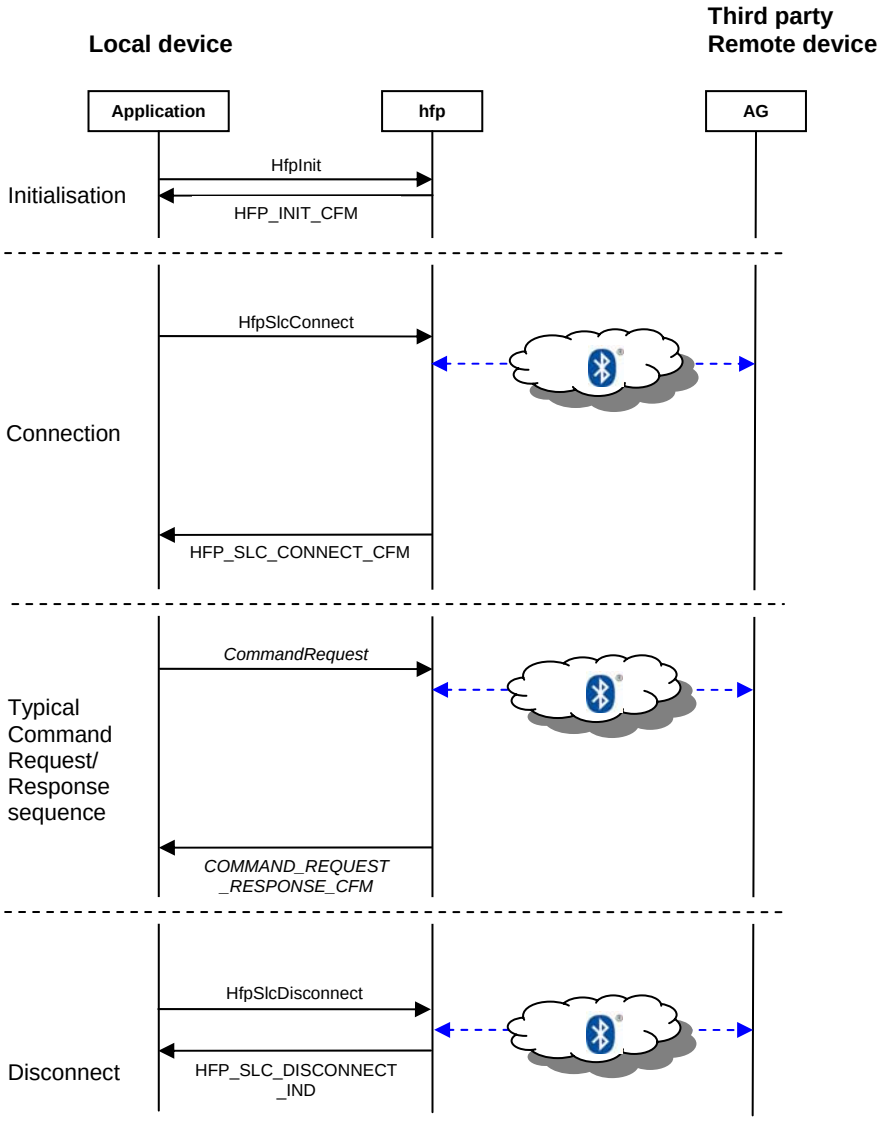
5.12 hfp.h

Interface to the Hands Free Profile library.

5.12.1 Purpose

This library implements the Hands Free Profile enabling the initiation, acceptance, handling and termination of calls.

5.12.2 Typical Message Sequence Chart



Note: Italic labels in the sequence chart indicate generic functions and messages, see the Reference Guide in the xIDE on-line help for a list of actual Hfp Command Requests.

An example of which would be an `HfpAnswerCall` (to answer an incoming call).

5.13 kalimba_standard_messages.h

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

5.13.1 Purpose

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

5.13.2 General

Users wishing to define their own kalimba messages should restrict the message ID to the range `0x000` to `0x669`. Values over this are reserved for kalimba_standard_messages. See the xIDE on-line Reference Guide for more details.

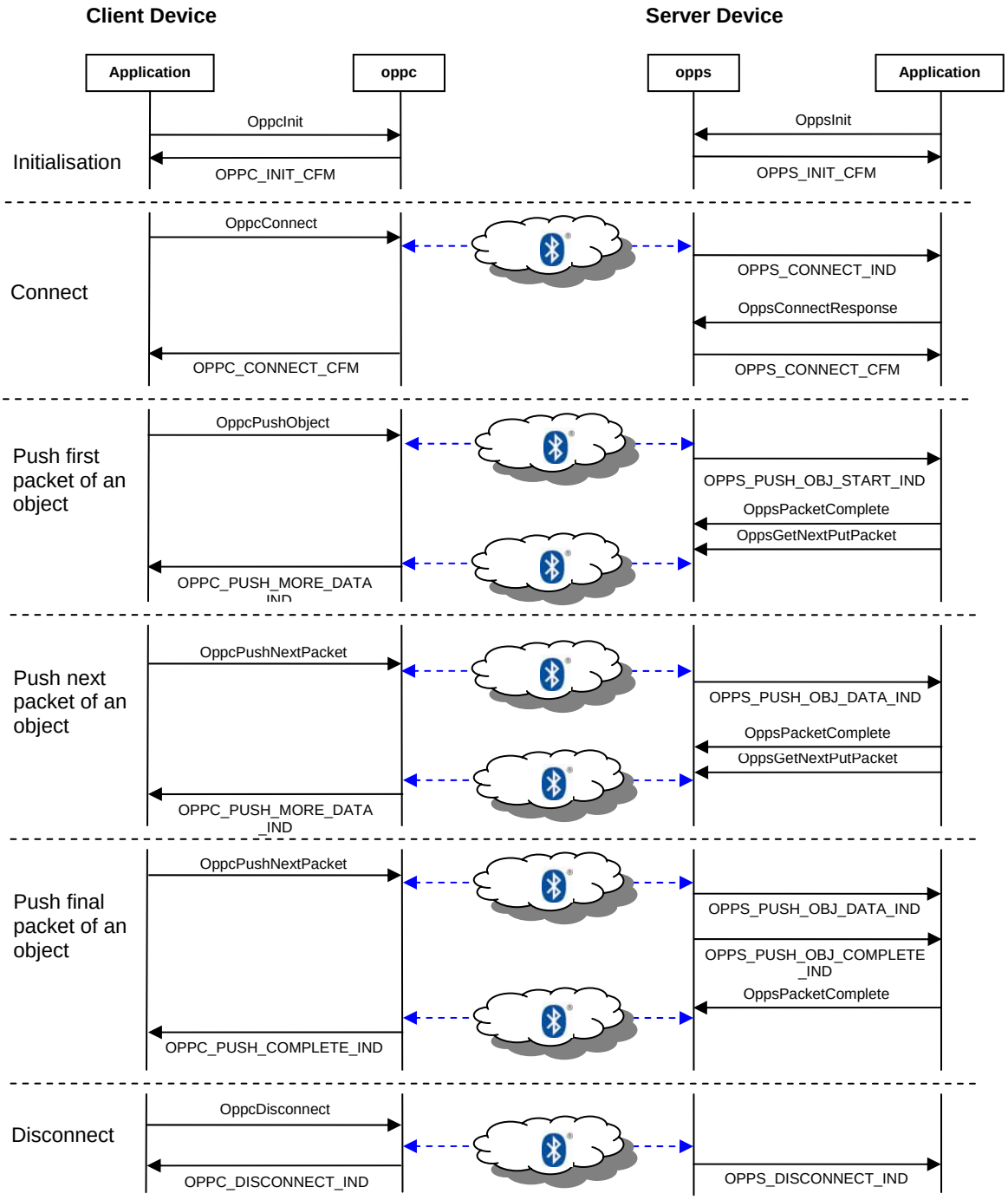
oppc.h

Interface to the OPP Client Profile library.

5.13.3 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.

5.13.4 Typical Message Sequence Chart



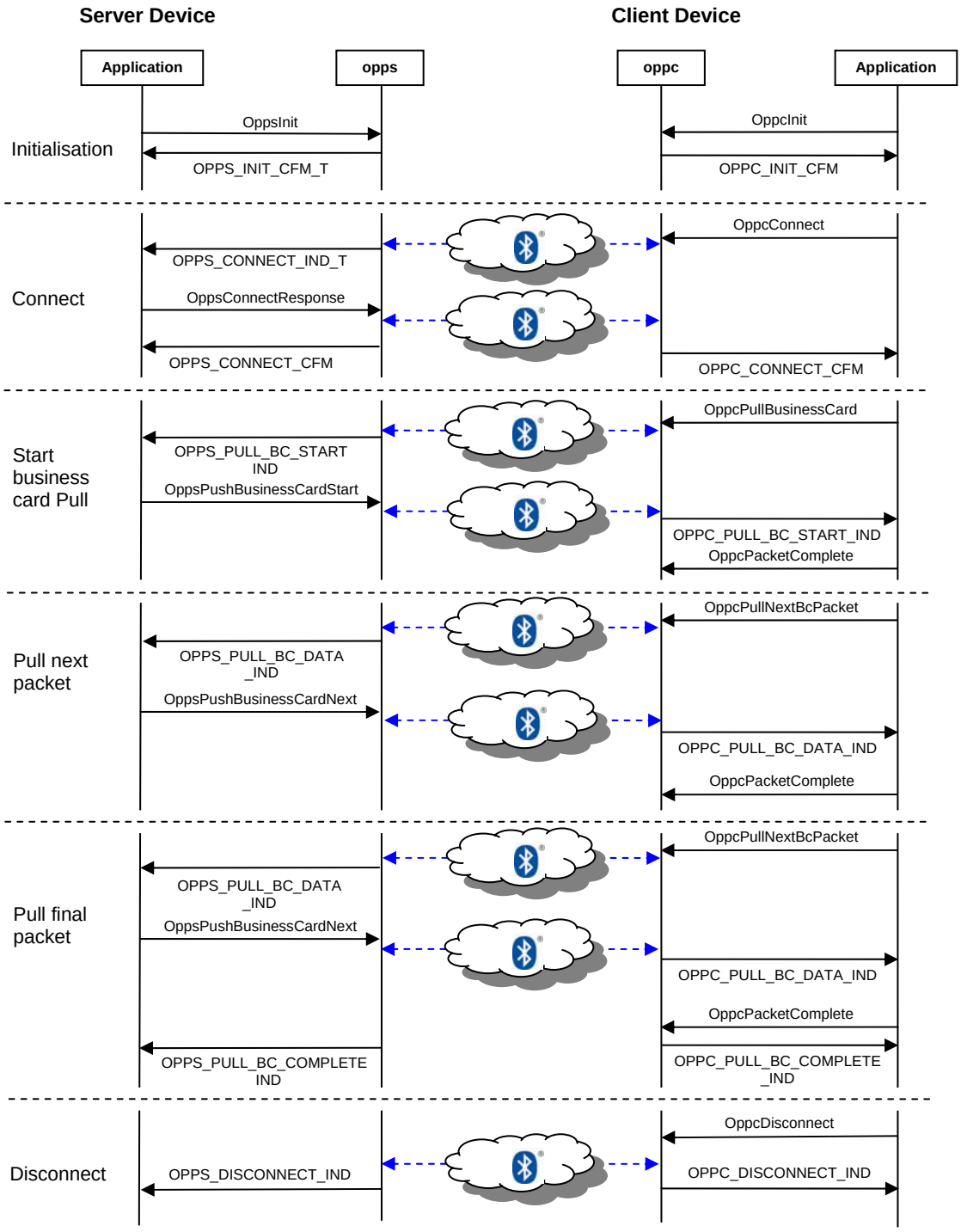
5.14 opps.h

Interface to the OPP Server Profile library.

5.14.1 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.

5.14.2 Typical Message Sequence Chart



5.15 print.h

Interface to the print library.

5.15.1 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:

5.15.2 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_ENABLE` 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.

5.16 region.h

Interface to the region library.

5.16.1 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.

5.16.2 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.

5.17 service.h

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

5.17.1 Purpose

This library provides functions that can used to 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.

5.17.2 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:

- `C:\BlueLab\src\lib\spp\spp_sdp.h`
- `C:\BlueLab\src\lib\hfp\hfp_sdp.h`

Where `C:\Bluelab` is the installation directory.

5.18 spp.h

Interface to the Serial Port Profile library.

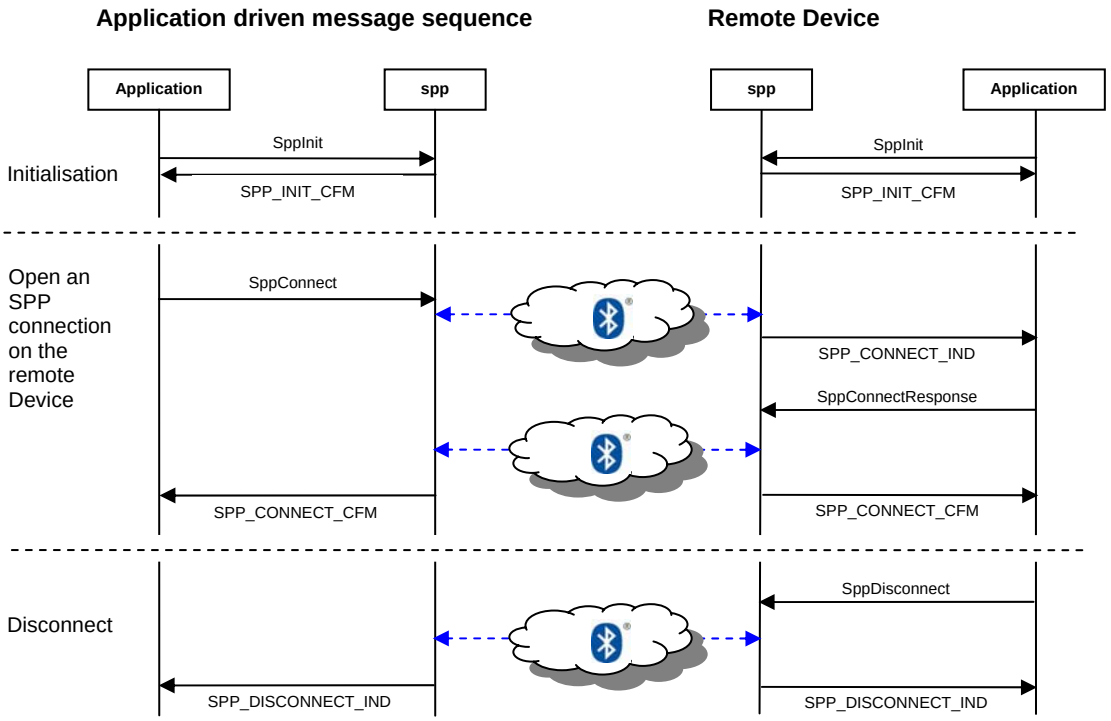
5.18.1 Purpose

This library implements the Serial Port Profile Specification. The Serial Port Profile Specification (hereafter referred to as 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.

As well as handling the RFCOMM link, the SPP library handles the registration of a service record with the SDP database, so that the service/applications can be reached via RFCOMM.

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

5.18.2 Typical Message Sequence Chart



6 Technical Support

Further information on all CSR products can be found on the technical support website (http://www.csrsupport.com).

Developers are also recommended to view the public newsgroups hosted by CSR on the Internet news (NTTP) server news.csr.com. The newsgroups are a convenient forum for the Bluetooth community to exchange knowledge and are a valuable source of information.

Set up instructions and guidelines for the use of newsgroups can be found by following the links on the CSR support website.

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
Bluetooth SIG	Bluetooth Special Interest Group
Casira	CSR Bluetooth development hardware
CSR	Cambridge Silicon Radio
A2DP	Advanced Audio Distribution Profile
ADC	Analogue to Digital Converter
API	Application Programming Interface
CT	Controller device
DAC	Digital to Analogue Converter
DM	Device Manager
DSP	Digital Signal Processor: a microprocessor dedicated to real-time signal processing.
FTP	File Transport Protocol
GAVDP	Generic Audio Visual Distribution Profile
GOEP	Generic Object Exchange Profile
HCI	Host Controller Interface
HFP	Hands Free Profile
L2CAP	Logical Link Controller and Adaptation Protocol
MMI	Man Machine Interface
OPP	Object Push Profile
PSM	Protocol/Service Multiplexer
RFCOMM	Serial Cable Emulation Protocol based on ETSI TS 07.10.
SCO	Synchronous Connection Orientation link
SDK	Software Development Kit
SDP	Service Discovery Protocol
SEP	Stream End Point
SNK	Data sink
SRC	Data source
TG	Target device
VM	Virtual Machine; environment in the BlueCore firmware for running application-specific code produced with BlueLab
xIDE	BlueLab's Integrated Development Environment

Document History

Revision	Date	Reason for Change
a	28 MAY 05	Original publication of this document. (CSR reference: blab-ug-003Pa)

BlueLab™

a guide to the BlueLab Libraries

User Guide

blab-ug-003Pa

May 2005