exactLE Software System Architecture

Document 2012-0023
Revision 1.1
September 25, 2015

Table of Contents

1 Introduction

This document describes the system architecture of Wicentric's Bluetooth low energy software system.

2 Software System

The Bluetooth LE software system consists of two main components, the *exactLE* Stack and *exactLE* Profiles. The *exactLE* Stack is complete protocol stack solution for single-mode Bluetooth LE devices. Wicentric's exactLE Profiles consists of sample applications, interoperable Bluetooth profile and service components, and a service layer for simplified application development and porting.

The software system is built on the Wicentric Software Foundation (WSF), an OS wrapper and porting layer. WSF also provides general-purpose software services such as queues, timers, and buffer management.

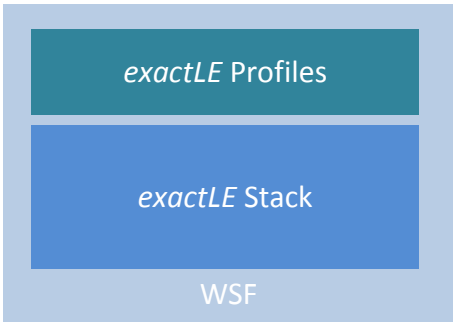

Figure 1. Wicentric's Bluetooth LE software system.

2.1 System Configuration

The *exactLE* Stack and Profiles are designed to support single-chip SoC systems and dual-chip systems. When operating in a single-chip system the *exactLE* Stack and Profiles run on the processor inside the SoC. A "thin" HCI layer adapts to the software interface of the target's LE link layer. When operating in a dual-chip system the *exactLE* Stack and Profiles run on a microcontroller and communicate with a Bluetooth LE controller chip over a wired interface such as UART or SPI. A standard transport-based HCI layer manages the communication between the two devices.

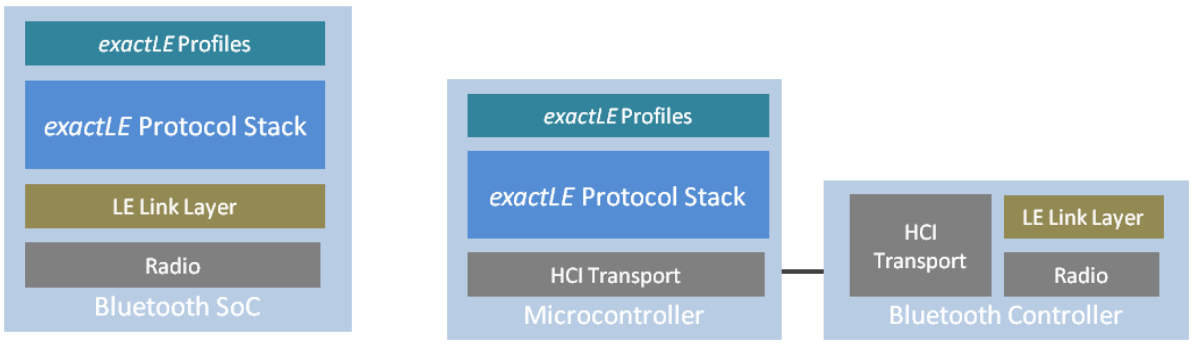

Figure 2. *exactLE* Stack and Profiles in a single-chip SoC system and dual-chip system.

2.2 *exactLE* Profiles

Wicentric's *exactLE* Profiles consist of sample applications, interoperable Bluetooth profile and service components, and a service layer called the App Framework for simplified application development and porting.

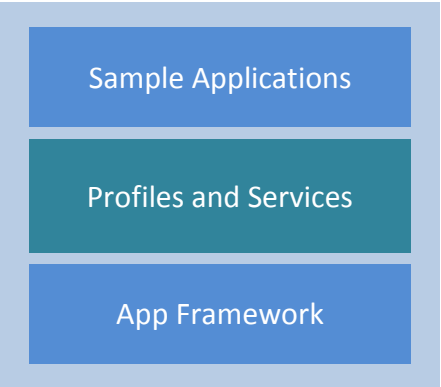

Figure 3. *exactLE* Profiles software system.

2.2.1 Sample Applications

Wicentric's Bluetooth low energy sample applications provide example source code for products such as a proximity keyfob, health sensor, and watch. The sample applications are designed with a product-oriented focus, with each application supporting one or more Bluetooth LE profile. The sample applications interface to the Profiles and Services and the App Framework.

2.2.2 Profiles and Services

The profiles and services are interoperable components designed to Bluetooth profile and service specification requirements. The profiles and services are used in applications to implement particular profile and service features.

The profiles are implemented in separate files for each profile role. The services, however, may be grouped together in files based on their logical function and the profile they are used by.

2.2.3 App Framework

The App Framework performs many operations common to Bluetooth LE embedded applications, such as:

- Application-level device, connection, and security management.
- Simple user interface abstractions for button press handling, sounds, display, and other user feedback.
- An abstracted device database for storing bonding data and other device parameters.

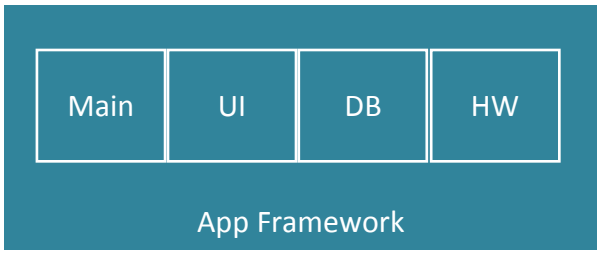

Figure 4. App Framework software subsystem.

The App Framework consists of several modules, each with their own API interface file:

- **Main**: Device, connection, and security management.
- **UI**: User interface abstraction.
- **DB**: Device database.
- **HW**: Hardware sensor interface abstraction.

2.3 *exactLE* Stack

The *exactLE* Stack is complete host protocol stack solution for single-mode Bluetooth LE devices. It consists of five protocol layers:

- **ATT**: Attribute protocol.
- **SMP**: Security manager protocol.
- **L2C**: L2CAP protocol.
- **HCI**: Host controller interface protocol.
- **DM**: Device manager.

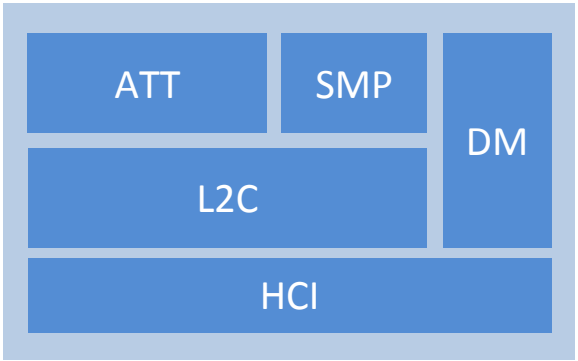

Figure 5. exactLE Stack software system.

2.3.1 ATT

The ATT subsystem implements the attribute protocol and generic attribute profile (GATT). It contains two independent subsystems: The attribute protocol client (ATTC) and attribute protocol server (ATTS).

ATTC implements all attribute protocol client features and is designed to meet the client requirements of the generic attribute profile. ATTC can support multiple simultaneous connections to different servers.

ATTS implements all attribute protocol server features and has support for multiple simultaneous client connections. ATTS also implements the server features defined by the generic attribute profile.

2.3.2 SMP

The SMP subsystem implements the security manager protocol. It contains two independent subsystems: The initiator (SMPI) and responder (SMPR).

SMPI implements the initiator features of the security manager protocol and has support for multiple simultaneous connections.

SMPR implements the responder features of the security manager protocol and has support for only one connection (by Bluetooth specification design).

SMP also implements the cryptographic toolbox, which uses AES. The interface to AES is asynchronous and abstracted through WSF. SMP also implements functions to support data signing.

2.3.3 L2C

The L2C subsystem implements the LE L2CAP protocol. It is a substantially scaled-down version of regular Bluetooth L2CAP.

In the TX data path, the main function of L2C is building L2CAP packets and sending them to HCI. In the RX data path, its main function is receiving packets from HCI and routing them to either SMP or ATT.

L2C also implements the connection parameter update procedure.

2.3.4 HCI

The HCI subsystem implements the host-controller interface specification. This specification defines commands, events, and data packets sent between a Bluetooth LE protocol stack on a host and a link layer on a controller.

The HCI API is optimized to be a thin interface layer for a single chip system. It is configurable for either a single chip system or traditional system with wired HCI.

This configurability is accomplished through a layered implementation. A core layer can be configured for either a single chip system or wired HCI. A transport and driver layer below the core layer can be configured for different wired transports such as UART.

2.3.5 DM

The DM subsystem implements device management procedures required by the stack. These procedures are partitioned by procedure category and device role (master or slave). The following procedures are implemented in DM:

- Advertising and device visibility: Enable/disable advertising, set advertising parameters and data, set connectability and discoverability.
- Scanning and device discovery: Start/stop scanning, set scan parameters, advertising reports, name discovery.
- Connection management: Create/accept/remove connections, set/update connection parameters, read RSSI.
- Security management: Bonding, storage of security parameters, authentication, encryption, authorization, random address management.
- Local device management: Initialization and reset, set local parameters, vendor-specific commands.

DM procedures support the Generic Access Profile (GAP) when applicable.

2.4 WSF

The Wicentric Software Foundation (WSF) is a simple OS wrapper, porting layer, and general-purpose software service used by the software system. The goal of WSF is to stay small and lean, supporting only the basic services required by the stack. It consists of the following:

- Event handler service with event and message passing.
- Timer service.
- Queue and buffer management service.
- Portable data types.
- Critical sections and task locking.
- Trace and assert diagnostic services.
- Security interfaces for encryption and random number generation.

3 Software Architecture

3.1 Interface Architecture

The software system uses function calls and callback functions in its APIs, as described below.

3.1.1 Message Passing API Functions

Message passing API functions result in a message being sent to the task running the stack. These functions typically involve a complex operation, such as creating a connection, and do not access internal (private) data.

3.1.2 Direct Execute API Functions

Direct execute API functions run entirely in the context of the calling function. These functions typically involve simple operations like reading or setting internal data. Task scheduling must be locked when accessing internal data.

3.1.3 Callback Functions

Callback functions are implemented by the client using the protocol stack and execute in the context of the stack. Callback functions are used to send events and data to the client.

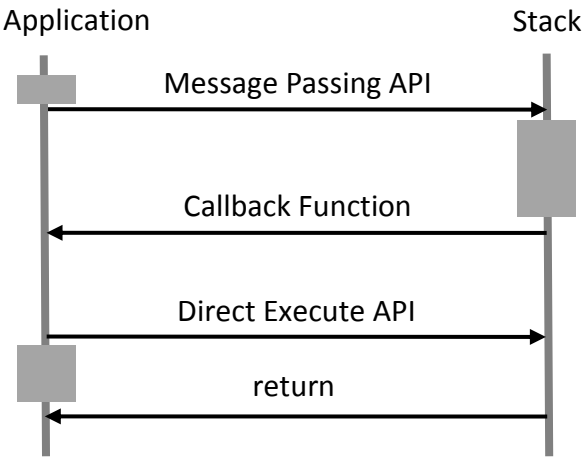

Figure 6. Message passing and direct execute interfaces.

3.2 Event Handlers and Tasks

The Wicentric software system defines an event handler service that forms a basis for the asynchronous communication mechanisms used in the system. An event handler can receive messages and events. Each software subsystem typically has its own event handler; for example, each layer of the protocol stack has its own event handler.

The stack is designed to be flexible and allow for different task architectures. The software system does not define any tasks but defines some interfaces to tasks. It relies on the target OS to implement tasks and manage the timer and event handler services from target OS tasks. A typical single-chip software system will use separate tasks for the application, stack, and link layer. However there is nothing in the design of the protocol stack or profiles that prevent them from being run in the same task as other software systems.

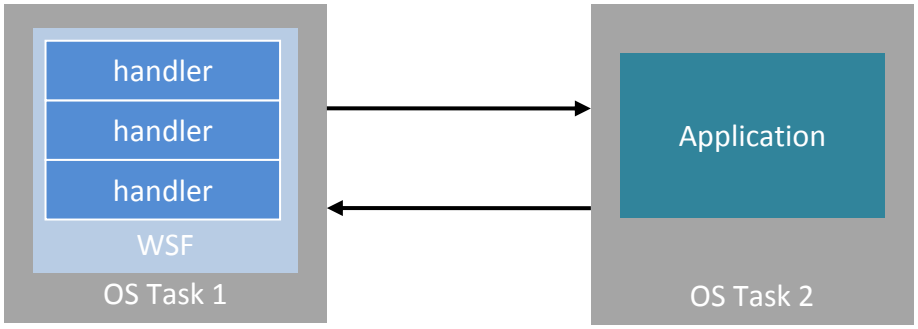

Figure 7. Example system showing event handlers executing within an OS task.

4 Data Path

4.1 TX Path

The TX data path covers the flow of data as it is sent from the application, through the stack, and then on to HCI.

There can be two data copies in the TX data path: when data is sent from the application to the stack and when data is sent from the stack to HCI. The stack does not copy data internally between layers.

The allocation and deallocation of data buffers takes place at the point where data is copied. When the application sends data to the stack, a buffer is allocated and data is copied to the buffer. When data is sent from the stack to the HCI or the link layer, the data is copied to an HCI or link layer buffer and the stack buffer is deallocated.

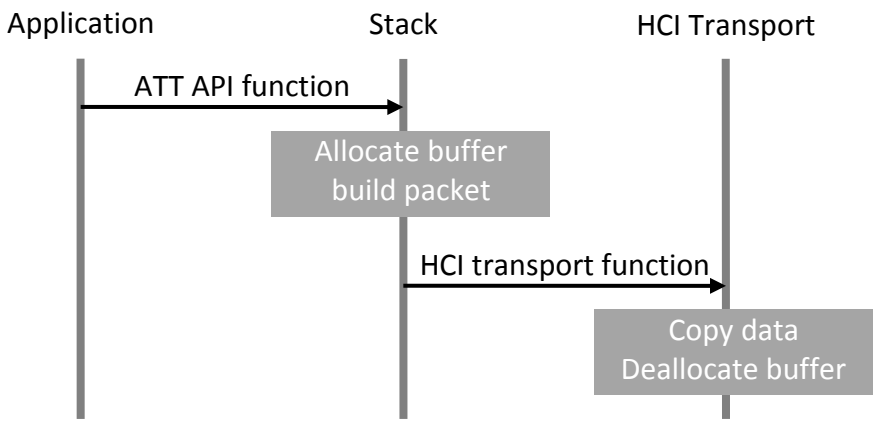

Figure 8. TX data path.

4.2 RX Path

The RX data path covers the flow of data as it is sent from HCI, through the stack, and then on to the application. Like the TX path, there can be two data copies in the RX data path: when data is sent from the stack to the application and when data is sent from HCI to the stack. The stack does not copy data internally between layers.

Buffers are allocated by the HCI layer and then deallocated internally by the stack.

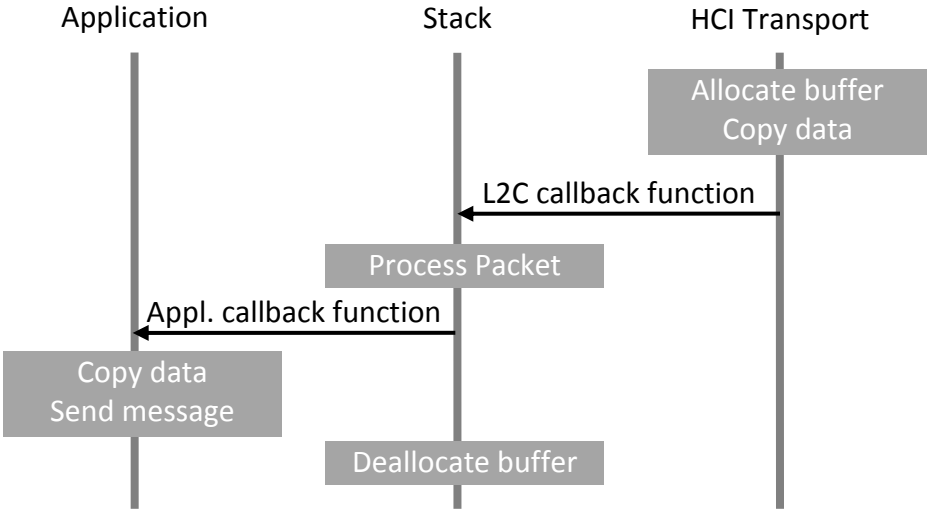

Figure 9. RX data path.

5 Porting

The porting process typically consists of two main steps:

1. Porting WSF interfaces and services to the target OS and software system.
2. Porting HCI to the target system and writing a transport driver, if applicable.

5.1 WSF Porting

Porting WSF typically consists of the following steps:

1. Create common data types for the target compiler.
2. Interface to a system timer to receive timer updates.
3. Implement WSF OS wrapper functions and interfaces.
4. Implement WSF diagnostics.

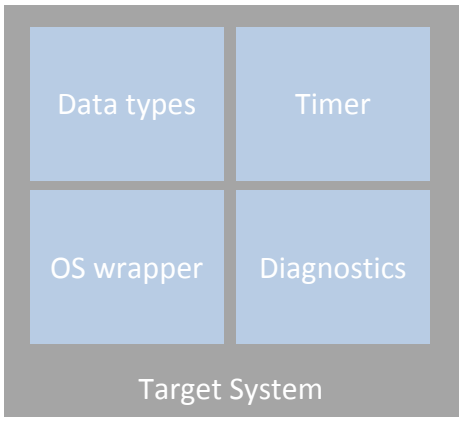

Figure 10. WSF porting process to a target system.

5.2 HCI Porting

Wicentric's HCI layer is designed to be portable and support different transport and chip configurations. The porting process depends on the chip configuration: If the stack is ported to a single-chip system then a "thin HCI" porting process is used. If the stack is ported to a two-chip system with wired HCI transport then a transport-based porting process is used.

6 Directory Structure

6.1 root

docs	Documentation
platform	Third party software for hardware platforms
projects	Project files for sample applications
sw	exactLE Stack and Profiles software

6.2 sw

apps	App Framework and sample applications
hci	HCI layer code
profiles	Bluetooth LE profiles
services	Bluetooth LE services
stack	Protocol stack
util	General purpose utilities
wsf	WSF

6.3 apps

apps	App Framework and sample applications
app	App Framework
datc	Proprietary data transfer client sample application
dats	Proprietary data transfer server sample application
fit	Fitness sensor sample application
gluc	Glucose sensor sample application
keyboard	HID keyboard sample application
medc	Health data collector sample application
meds	Health sensor sample application
mouse	HID mouse sample application
remote	HID remote sample application
tag	Proximity tag sample application
watch	Watch sample application

6.4 profiles

profiles	Bluetooth LE profiles
anpc	Alert Notification Profile client
bas	Battery Service server
blpc	Blood Pressure Profile client
blps	Blood Pressure Profile server
dis	Device Information Service client
fmpl	Find Me Profile locator
gatt	Generic Attribute Profile client
glpc	Glucose Profile client
glps	Glucose Profile server
hid	HID device
hrpc	Heart Rate Profile client
hrps	Heart Rate Profile server
htpc	Health Thermometer Profile client
htps	Health Thermometer Profile server

paspc	Phone Alert Status Profile client
tipc	Time Profile client
wpc	Wicentric proprietary profile client
wspc	Weight Scale Profile client
wsps	Weight Scale Profile server

7 Documentation

Software System	2012_0023_exactLE_Software_System.pdf
Profiles, App Framework, Sample Applications	2011_0020_App_Framework_API.pdf 2012_0021_Profile_and_Service_API.pdf 2012_0022_Sample_App_Users_Guide.pdf
Stack API	2009_0006_HCI_API.pdf 2009_0007_L2C_SDD.pdf 2009_0008_Device_Manager_API.pdf 2009_0010_Attribute_Protocol_API.pdf
WSF API	2009_0003_WSF_API.pdf
Porting Guide	2010_0014_Stack_Porting_Guide.pdf

8 Definitions

ATT	Attribute Protocol, also attribute protocol software subsystem
ATTC	Attribute protocol client software subsystem
ATTS	Attribute protocol server software subsystem
DM	Device Manager software subsystem
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HCI	Host Controller Interface
L2C	L2CAP software subsystem
L2CAP	Logical Link Control Adaptation Protocol
LE	(Bluetooth) Low Energy
SMP	Security Manager Protocol, also security manager protocol software subsystem
SMPI	Security Manager Protocol Initiator software subsystem
SMPR	Security Manager Protocol Responder software subsystem
WSF	Wicentric Software Foundation software service and porting layer.