# L2CAP Software Design Document

Document 2009-0007
Revision 1.2
September 25, 2015

# Table of Contents

1 Introduction ..... 3
2 System Context ..... 3
3 Subsystem Architecture ..... 3
3.1 l2c_api ..... 4
3.1.1 Constants and Data Structures ..... 4
3.1.2 Function Calls ..... 4
3.1.3 Callback Functions ..... 7
3.2 l2c_main ..... 7
3.2.1 Constants and Data Structures ..... 8
3.2.2 Functions ..... 8
3.3 l2c_master ..... 9
3.3.1 Functions ..... 9
3.4 l2c_slave ..... 9
3.4.1 Constants and Data Structures ..... 9
3.4.2 Functions ..... 10
4 Scenarios ..... 10
4.1 Initialization ..... 10
4.2 Data Path ..... 10
4.3 Connection Parameter Update ..... 11
5 References ..... 12
6 Definitions ..... 12

# 1 Introduction

This document describes the API and software design of the L2CAP subsystem, L2C.

# 2 System Context

Figure 1 shows the context of the L2C subsystem in the Bluetooth LE stack.

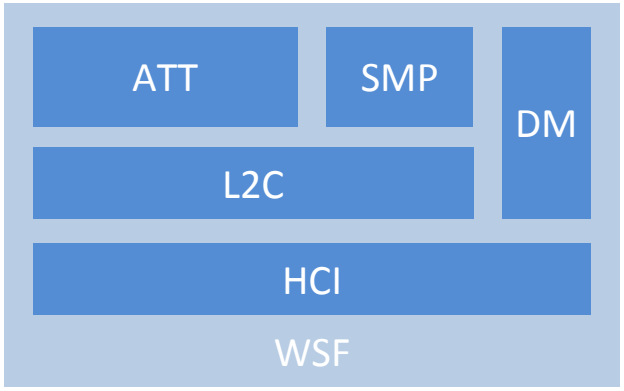


**Figure 1. Bluetooth LE stack software system.**

L2C interfaces to HCI to send and receive ACL packets. The ATT and SMP protocol layers interface to L2C to send and receive L2CAP packets. L2C also interfaces to DM to perform the L2CAP connection update procedure.

# 3 Subsystem Architecture

Figure 2 shows the different modules that make up the L2C subsystem.


**Figure 2. Subsystem architecture**

Module l2c_api contains the API. Module l2c_main contains the main API function implementation, main event handler, and functions for processing packets. Module l2c_master contains API functions and other functions used only when operating as an LE master. Module l2c_slave contains API functions

and other functions used only when operating as an LE slave. Module lcc_coc contains functions for L2CAP Connection Oriented Channels.

## 3.1 l2c_api

### 3.1.1 Constants and Data Structures

#### 3.1.1.1 Connection Identifiers

| Name | Value | Description |
|---|---|---|
| L2C_CID_ATT | 0x0004 | CID for attribute protocol. |
| L2C_CID_LE_SIGNALING | 0x0005 | CID for LE signaling. |
| L2C_CID_SMP | 0x0006 | CID for security manager protocol. |

#### 3.1.1.2 Connection Parameter Result

| Name | Value | Description |
|---|---|---|
| L2C_CONN_PARAM_ACCEPTED | 0x0000 | Connection parameters accepted. |
| L2C_CONN_PARAM_REJECTED | 0x0001 | Connection parameters rejected. |

#### 3.1.1.3 Control Callback Events

| Name | Value | Description |
|---|---|---|
| L2C_CTRL_FLOW_ENABLE_IND | 0x00 | Data flow enabled. The client may call L2cDataReq(). |
| L2C_CTRL_FLOW_DISABLE_IND | 0x01 | Data flow disabled. The client should not call L2cDataReq() until it receives a L2C_CTRL_FLOW_ENABLE_IND. |

### 3.1.2 Function Calls

#### 3.1.2.1 void L2cInit(void)

This function is called to initialize L2C. This function is generally called once during system initialization before any other non-initialization L2C API functions are called.

#### 3.1.2.2 void L2cMasterInit(void)

This function is called to initialize L2C for operation as a Bluetooth LE master. This function is generally called once during system initialization before any other non-initialization L2C API functions are called.

#### 3.1.2.3 void L2cSlaveInit(void)

This function is called to initialize L2C for operation as a Bluetooth LE slave. This function is generally called once during system initialization before any other non-initialization L2C API functions are called.

#### 3.1.2.4 void L2cRegister(uint16_t cid, l2cDataCback_t dataCback, l2cCtrlCback_t ctrlCback)

This function is called by the L2C client, such as ATT or SMP, to register for the given CID. This allows the client to send and receive data using that CID.

- **dataCback**: Callback function for L2CAP data received for this CID. This cannot be set to NULL.

- **ctrlCback**: Callback function for control events for this CID. This cannot be set to NULL.

This function stores the callback parameters in l2cMain.

#### 3.1.2.5 void L2cDataReq(uint16_t cid, uint16_t handle, uint16_t len, uint8_t *pL2cPacket)

This function sends an L2CAP data packet on the given CID.

- **cid**: The channel identifier.
- **handle**: The connection handle. The client receives this handle from DM when the connection is established.
- **len**: The length of the payload data in pPacket.
- **pL2cPacket:** A buffer containing the packet. This is a WSF buffer allocated by the client.

The buffer pointed to by pL2cPacket must be a WSF buffer allocated by the client.

This function first checks if there is an active connection associated with the handle. If not, the packet is discarded and the buffer containing the packet is deallocated. Then it builds an L2CAP data packet, setting both the L2CAP and HCI headers. Then it calls function HciSendAclData() to send the packet to HCI.

#### 3.1.2.6 void L2cDmConnUpdateReq(uint16_t handle, hciConnSpec_t *pConnSpec)

This function is called by DM to send an L2CAP connection update request.

- **handle**: The connection handle.
- **pConnSpec**: Pointer to the connection specification structure. This structure is defined in [1]. The following elements in the structure must be set:
  - connIntervalMin
  - connIntervalMax
  - connLatency
  - supTimeout

This function starts the signaling request timeout timer, builds an L2CAP connection update request packet and then calls L2cDataReq() to send the packet.

#### 3.1.2.7 void L2cDmConnUpdateRsp(uint8 identifier, uint16_t handle, uint16_t result)

This function is called by DM to send an L2CAP connection update response.

- **identifier**: Identifier value previously passed from L2C to DM.
- **handle**: The connection handle.
- **result**: Connection update response result. See 3.1.1.2.

This function builds an L2CAP connection update response packet and then calls L2cDataReq() to send the packet.

#### 3.1.2.8 L2cSlaveHandler(wsfEventMask_t event, wsfMsgHdr_t *pMsg)

This function is the WSF event handler for L2C when operating as a slave. This function is only called from the WSF OS implementation.

- **event**: Event mask.
- **pMsg**: Pointer to message.

The implementation of this function handles the L2CAP signaling request timeout timer.

#### 3.1.2.9 L2cSlaveHandlerInit(wsfHandlerId_t handlerId)

This is the event handler initialization function for L2C when operating as a slave. This function is generally called once during system initialization.

- **handlerId**: ID for this event handler.

This function stores the hander ID and performs other initialization procedures.

#### 3.1.2.10 L2cCocInit(void)

This function initializes the L2Cap Connection Oriented Channels. This function is generally called once during initialization.

#### 3.1.2.11 l2cCocRegId_t L2cCocRegister(l2cCocCback_t cback, l2cCocReg_t *pReg)

This function is used to register an instance of a connection oriented channel. The instance can be a channel acceptor, initiator, or both. If registering as channel as acceptor, then the PSM is specified. After registering a connection, the connections can be established by the client using this registration instance.

- **cback**: Callback for the connection oriented channel.
- **pReg**: Registration parameters.

This function returns an identifier for the channel.

#### 3.1.2.12 L2cCocDeregister(l2cCocRegId_t regId)

This function deregisters and deallocates a connection oriented channel registered instance. This function should only be called if there are no active channels using the registration instance.

- **regId**: The identifier for the channel (returned by L2cCocRegister).

#### 3.1.2.13 uint16_t L2cCocConnectReq(dmConnId_t connId, l2cCocRegId_t regId, uint16_t psm)

This function initiates a connection to the given peer PSM using the connection oriented channel subsystem.

- **connId:** The DM connection ID.
- **regId:** The identifier for the channel (returned by L2cCocRegister).
- **psm:** The peers PSM.

This function returns the local CID or L2C_COC_CID_NONE if there was a failure.

#### 3.1.2.14 L2cCocDisconnectReq(uint16_t cid)

This function disconnects the channel to the peer for the given CID.

- **cid:** The channel CID (returned by L2cCocConnectReq).

#### 3.1.2.15 L2cCocDataReq(uint16_t cid, uint16_t len, uint8_t *pPayload)

This function sends an L2CAP data packet on the given connection oriented channel with the given CID.

- **cid:** The channel CID (returned by L2cCocConnectReq).
- **len:** The length of the pPayload in bytes.
- **pPayload:** The packet to send.

### 3.1.3 Callback Functions

#### 3.1.3.1 void (*l2cDataCback_t)(uint16_t handle, uint16_t len, uint8_t *pPacket)

This callback function sends a received L2CAP packet to the client.

- **handle**: The connection handle.
- **len**: The length of the L2CAP payload data in pPacket.
- **pPacket:** A buffer containing the packet.

#### 3.1.3.2 void(*l2cCtrlCback_t)(uint8_t event)

This callback function sends control events to the client. It is currently used only for flow control.

- **event**: Control event. See 3.1.1.3

#### 3.1.3.3 void (*l2cCocCback_t)(l2cCocEvt_t *pMsg)

This callback function sends data and other events to connection oriented channel clients.

- **pMsg:** Pointer to the message structure

#### 3.1.3.4 uint16_t (*l2cCocAuthorCback_t)(dmConnId_t connId, l2cCocRegId_t regId, uint16_t psm)

This callback function is used for authorization of connection oriented channels.

- **connId**: The connection identifier.
- **regId**: The connection oriented channel registration instance identifier.
- **psm**: The psm of the connection.

## 3.2 l2c_main

This module implements the main processing functions of L2C which are common for both master and slave. It also contains the API functions listed above which are common for both master and slave.

### 3.2.1 Constants and Data Structures

#### 3.2.1.1 l2cMain_t

| Type | Name | Description |
|---|---|---|
| l2cDataCback_t | attDataCback | Data callback for ATT. |
| l2cDataCback_t | smpDataCback | Data callback for SMP. |
| l2cCtrlCback_t | attCtrlCback | Control callback for ATT. |
| l2cCtrlCback_t | smpCtrlCback | Control callback for SMP. |
| l2cDataCback_t | masterRxSignalingPkt | Master signaling packet processing function. |
| l2cDataCback_t | slaveRxSignalingPkt | Slave signaling packet processing function. |
| l2cDataCidCback_t | l2cDataCidCback | Data callback for L2CAP on other CIDs |

### 3.2.2 Functions

#### 3.2.2.1 l2cHciAclCback(uint8 *pPacket)

This is the HCI ACL data callback function. It is the main receive packet processing function and operates according to the following pseudocode:

```
parse HCI handle and length
verify handle
verify length

parse L2CAP length
verify L2CAP length vs HCI length

parse cid
switch cid:
  case L2C_CID_LE_SIGNALING
    l2cCb.l2cSignalingCback()
    break
case L2C_CID_ATT
    l2cCb.attDataCback()
    break
case L2C_CID_SMP
    l2cCb.smpDataCback()
    break
default
    l2cCb.l2cDataCidCback()

deallocate buffer
```

#### 3.2.2.2 l2cRxSignalingPkt(uint16_t handle, uint16_t len, uint8_t *pPacket)

This function processes received L2CAP signaling packets. It operates according to the following pseudocode:

```
role = DmGetRole(handle)
if role == master and configured for master
  call l2cMain.masterRxSignalingPkt()
else if role == slave and configured for slave
  call l2cMain.slaveRxSignalingPkt()
else
  warning message
```

#### 3.2.2.3 l2cSendCmdReject(uint16_t handle, uint8_t identifier, uint16_t reason, uint16_t param)

This function builds and sends a command reject message. It allocates a buffer, builds the message, and calls L2cDataReq() to send the message.

## 3.3 l2c_master

This module implements functions specific to the master, API functions.

### 3.3.1 Functions

#### 3.3.1.1 l2cMasterRxSignalingPkt(uint16_t handle, uint8_t *pPacket)

This function processes L2CAP signaling packets when operating as a master. It operates according to the following pseudocode:

```
parse code, len, and identifier
if code != conn param update req or
   len != len for conn param update req
   l2cSendCmdReject()
   return
else if identifier == 0
   return
parse parameters
check parameter range; if invalid,
  L2cDmConnUpdateRsp(identifier, handle, L2C_CONN_PARAM_REJECTED)
  return
DmL2cConnUpdateInd()
```

## 3.4 l2c_slave

This module implements functions specific to the slave, including API functions.

### 3.4.1 Constants and Data Structures

#### 3.4.1.1 l2cSlave_t

| Type | Name | Description |
|---|---|---|
| wsfTimer_t | reqTimer | Signaling request timeout timer. |
| wsfHandlerId_t | handerId | ID for this event handler. |
| uint8_t | identifier | Signaling request identifier. |

### 3.4.2 Functions

#### 3.4.2.1 l2cSlaveRxSignalingPkt(uint16_t handle, uint8_t *pPacket)

This function processes L2CAP signaling packets when operating as a slave. It operates according to the following pseudocode:

```
parse code, len, and identifier
if code != conn param update rsp or
len != len for conn param update rsp
  l2cSendCmdReject()
  return
if identifier != l2cSlave.identifier
   return
parse parameters
DmL2cConnUpdateCnf()
```

# 4 Scenarios

## 4.1 Initialization

Figure 3 shows the initialization process. In this example, the system supports operation as both a master and a slave so L2cMasterInit() and L2cSlaveInit() are called. Then function L2cSlaveHandlerInit() is called after L2cSlaveHandler() is set up in the WSF OS implementation.

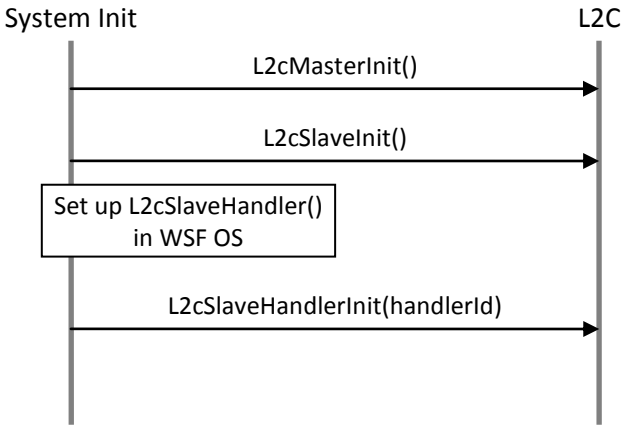


**Figure 3. Initialization.**

## 4.2 Data Path

Figure 4 shows the operation of the data path with ATT shown as an example L2C client. ATT calls L2cDataReq() to send a packet to L2C. Then L2C calls HciSendAclData to send the packet to HCI. In the

receive direction, HCI calls HciAclDataCback() to send a packet to L2C. L2C calls ATT callback function attDataCback() to send the packet to ATT.

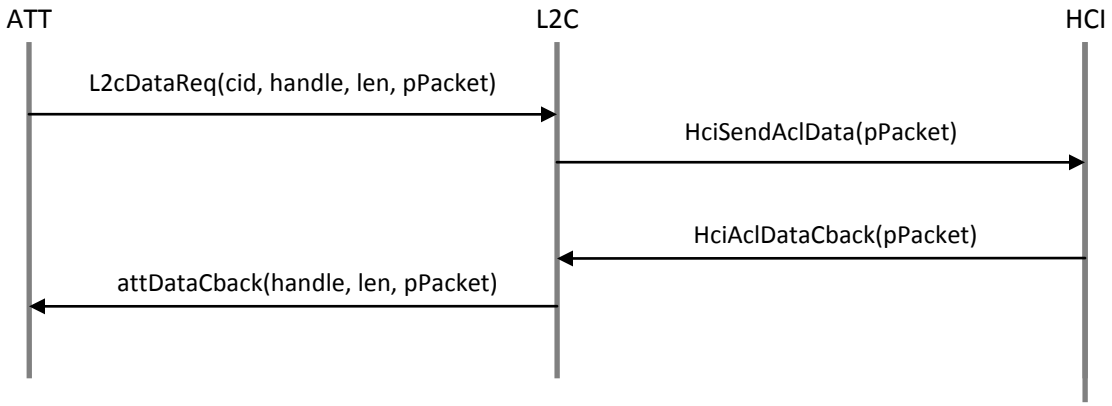


**Figure 4. Data path.**

## 4.3 Connection Parameter Update

Figure 5 shows a connection parameter update procedure with the stack operating as a slave. DM calls L2cDmConnUpdateReq() to initiate the process. L2C builds and sends an L2CAP Connection Parameter Update Request. The peer device receives the request and initiates a connection update procedure. When the procedure completes an HCI LE Connection Update Complete Event is sent from HCI to DM. Then the L2CAP Connection Parameter Update Response is received from the peer and L2C calls DmL2cConnUpdateCnf().

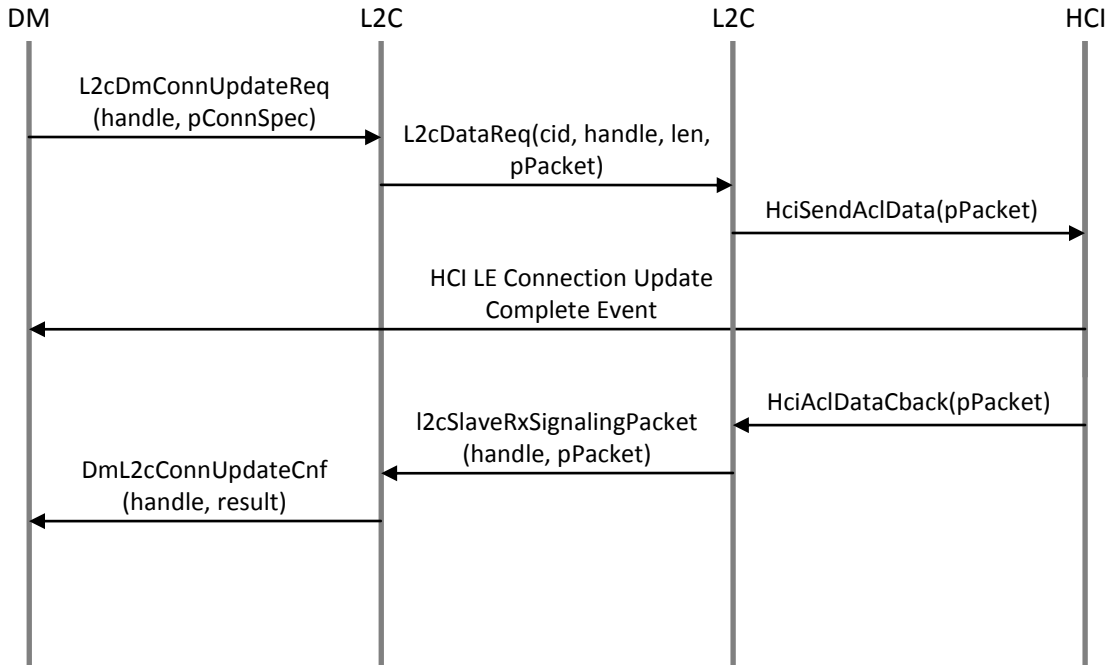


**Figure 5. Connection Parameter Update.**

# 5 References

1. Wicentric, “HCI API”, 2009-0006.
2. Wicentric, “Software Foundation API”, 2009-0003.

# 6 Definitions

| | |
|---|---|
| ACL | Asynchronous Connectionless data packet |
| CID | Connection Identifier |
| DM | Device Manager software subsystem |
| HCI | Host Controller Interface |
| L2C | L2CAP software subsystem |
| L2CAP | Logical Link Control Adaptation Protocol |
| LE | (Bluetooth) Low Energy |
| SMP | Security Manager Protocol |
| WSF | Wicentric Software Foundation software service and porting layer |