Part D

LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION

This document describes the Bluetooth logical link control and adaptation protocol (L2CAP). This protocol supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. This document is part of the Bluetooth Specification. This document describes the protocol state machine, packet format and composition, and a test interface required for the Bluetooth test and certification program.

CONTENTS

1 INTRODUCTION

This section of the Bluetooth Specification defines the Logical Link Control and Adaptation Layer Protocol, referred to as L2CAP. L2CAP is layered over the Baseband Protocol and resides in the data link layer as shown in Figure 1.1. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

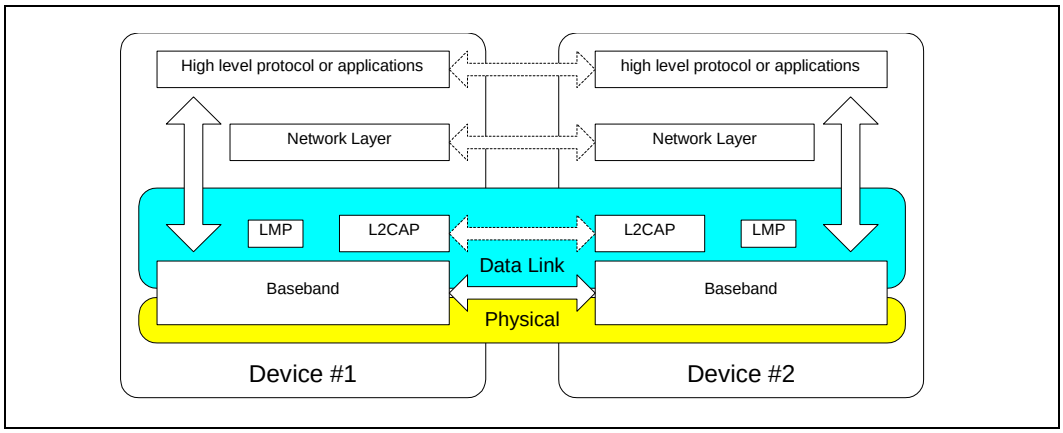

Figure 1.1: L2CAP within protocol layers

The "Baseband Specification" on page 33 defines two link types: Synchronous Connection-Oriented (SCO) links and Asynchronous Connection-Less (ACL) links. SCO links support real-time voice traffic using reserved bandwidth. ACL links support best effort traffic. The L2CAP Specification is defined for only ACL links and no support for SCO links is planned.

For ACL links, use of the AUX1 packet on the ACL link is prohibited. This packet type supports no data integrity checks (no CRC). Because L2CAP depends on integrity checks in the Baseband to protect the transmitted information, AUX1 packets must never be used to transport L2CAP packets.

The format of the ACL payload header is shown below. Figure 1.2 on page 250 displays the payload header used for single-slot packets and Figure 1.3 displays the header used in multi-slot packets. The only difference is the size of the length field. The packet type (a field in the Baseband header) distinguishes single-slot packets from multi-slot packets.

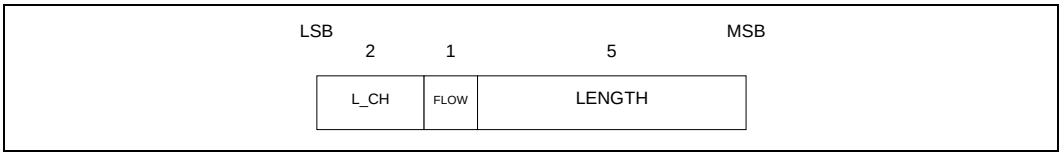

Figure 1.2: ACL Payload Header for single-slot packets

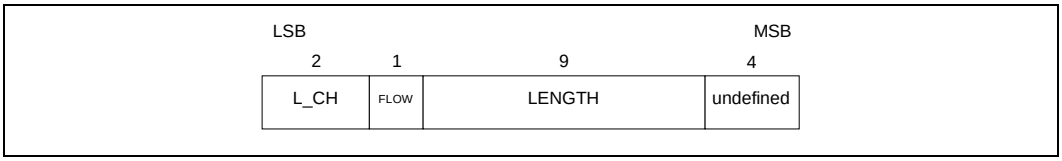

Figure 1.3: ACL Payload Header for multi-slot packets

The 2-bit logical channel (L_CH) field, defined in Table 1.1, distinguishes L2CAP packets from Link Manager Protocol (page 185) packets. The remaining code is reserved for future use.

L_CH code	Logical Channel	Information
00	RESERVED	Reserved for future use
01	L2CAP	Continuation of L2CAP packet
10	L2CAP	Start of L2CAP packet
11	LMP	Link Manager Protocol

Table 1.1: Logical channel L_CH field contents

The FLOW bit in the ACL header is managed by the Link Controller (LC), a Baseband implementation entity, and is normally set to1 (“flow on”). It is set to 0 (“flow off”) when no further L2CAP traffic should be sent over the ACL link. Sending an L2CAP packet with the FLOW bit set to 1 resumes the flow of incoming L2CAP packets. This is described in more detail in “Baseband Specification” on page 33.

1.1 L2CAP FUNCTIONAL REQUIREMENTS

The functional requirements for L2CAP include protocol multiplexing, segmentation and reassembly (SAR), and group management. Figure 1.4 illustrates how L2CAP fits into the Bluetooth Protocol Stack. L2CAP lies above the Baseband Protocol (page 33) and interfaces with other communication protocols such as the Bluetooth Service Discovery Protocol (SDP, page 321), RFCOMM (page 383), and Telephony Control (TCS, page 427). Voice-quality channels for audio and telephony applications are usually run over Baseband SCO links. Packetised audio data, such as IP Telephony, may be sent using communication protocols running over L2CAP.

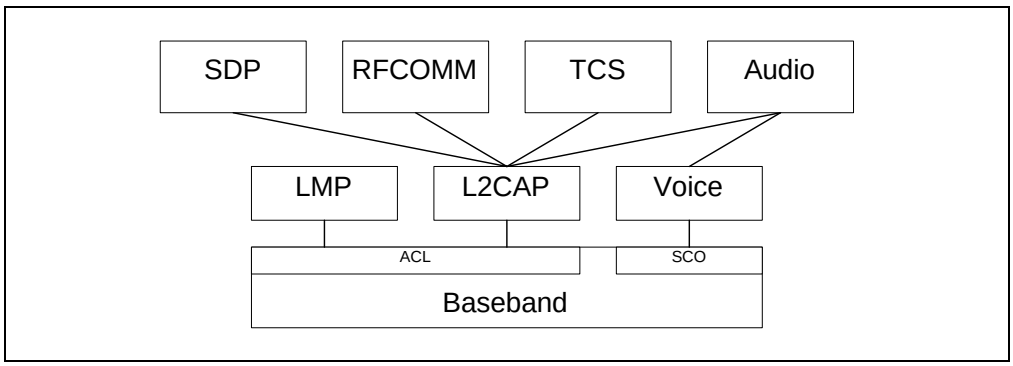

Figure 1.4: L2CAP in Bluetooth Protocol Architecture

Essential protocol requirements for L2CAP include simplicity and low overhead. Implementations of L2CAP must be applicable for devices with constrained computational resources. L2CAP should not consume excessive power since that significantly sacrifices power efficiency achieved by the Bluetooth Radio. Memory requirements for protocol implementation should also be kept to a minimum.

The protocol complexity should be acceptable to personal computers, PDAs, digital cellular phones, wireless headsets, joysticks and other wireless devices supported by Bluetooth. Furthermore, the protocol should be designed to achieve reasonably high bandwidth efficiency.

- *Protocol Multiplexing*

  L2CAP must support protocol multiplexing because the Baseband Protocol does not support any “type” field identifying the higher layer protocol being multiplexed above it. L2CAP must be able to distinguish between upper layer protocols such as the Service Discovery Protocol (page 321), RFCOMM (page 383), and Telephony Control (page 427).

- *Segmentation and Reassembly*

  Compared to other wired physical media, the data packets defined by the Baseband Protocol (page 33) are limited in size. Exporting a maximum transmission unit (MTU) associated with the largest Baseband payload (341 bytes for DH5 packets) limits the efficient use of bandwidth for higher layer protocols that are designed to use larger packets. Large L2CAP packets must be segmented into multiple smaller Baseband packets prior to their transmission over the air. Similarly, multiple received Baseband packets may be reassembled into a single larger L2CAP packet following a simple integrity check (described in Section 2.4.2 on page 256). The Segmentation and Reassembly (SAR) functionality is absolutely necessary to support protocols using packets larger than those supported by the Baseband.

- *Quality of Service*

  The L2CAP connection establishment process allows the exchange of information regarding the quality of service (QoS) expected between two Blue-

tooth units. Each L2CAP implementation must monitor the resources used by the protocol and ensure that QoS contracts are honoured.

- *Groups*

  Many protocols have the concept of a group of addresses. The Baseband Protocol supports the concept of a piconet, a group of devices synchronously hopping together using the same clock. The L2CAP group abstraction permits implementations to efficiently map protocol groups on to piconets. Without a group abstraction, higher level protocols would need to be exposed to the Baseband Protocol and Link Manager functionality in order to manage groups efficiently.

1.2 ASSUMPTIONS

The protocol is designed based on the following assumptions:

1. The ACL link between two units is set up using the Link Manager Protocol (page 185). The Baseband provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than 1 ACL link exists between any two devices.
2. The Baseband always provides the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bi-directional. Multicasts and unidirectional traffic (e.g., video) do not require duplex channels.
3. L2CAP provides a reliable channel using the mechanisms available at the Baseband layer. The Baseband always performs data integrity checks when requested and resends data until it has been successfully acknowledged or a timeout occurs. Because acknowledgements may be lost, timeouts may occur even after the data has been successfully sent. The Baseband protocol uses a 1-bit sequence number that removes duplicates. Note that the use of Baseband broadcast packets is prohibited if reliability is required since all broadcasts start the first segment of an L2CAP packet with the same sequence bit.

1.3 SCOPE

The following features are outside the scope of L2CAP's responsibilities:

- L2CAP does not transport audio designated for SCO links.
- L2CAP does not enforce a reliable channel or ensure data integrity, that is, L2CAP performs no retransmissions or checksum calculations.
- L2CAP does not support a reliable multicast channel. See Section 4.2.
- L2CAP does not support the concept of a global group name.

2 GENERAL OPERATION

The Logical Link Control and Adaptation Protocol (L2CAP) is based around the concept of *“channels”*. Each one of the end-points of an L2CAP channel is referred to by a *channel identifier.*

2.1 CHANNEL IDENTIFIERS

Channel identifiers (CIDs) are local names representing a logical channel end-point on the device. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. The null identifier (0x0000) is defined as an illegal identifier and must never be used as a destination end-point. Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that the same CID is not reused as a local L2CAP channel endpoint for multiple simultaneous L2CAP channels between a local device and some remote device. Table 2.1 summarises the definition and partitioning of the CID name space.

CID assignment is relative to a particular device and a device can assign CIDs independently from other devices (unless it needs to use any of the reserved CIDs shown in the table below). Thus, even if the same CID value has been assigned to (remote) channel endpoints by several remote devices connected to a single local device, the local device can still uniquely associate each remote CID with a different device.

CID	Description
0x0000	Null identifier
0x0001	Signalling channel
0x0002	Connectionless reception channel
0x0003-0x003F	Reserved
0x0040-0xFFFF	Dynamically allocated

Table 2.1: CID Definitions

2.2 OPERATION BETWEEN DEVICES

Figure 2.1 on page 254 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID identifies each endpoint of the channel. The connectionless channels restrict data flow to a single direction. These channels are used to support a channel “group” where the CID on the source represents one or more remote devices. There are also a number of CIDs reserved for special purposes. The signalling channel is one example of a reserved channel. This channel is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of these channels. Support for a signalling chan-

nel within an L2CAP entity is mandatory. Another CID is reserved for all incoming connectionless data traffic. In the example below, a CID is used to represent a group consisting of device #3 and #4. Traffic sent from this channel ID is directed to the remote channel reserved for connectionless data traffic.

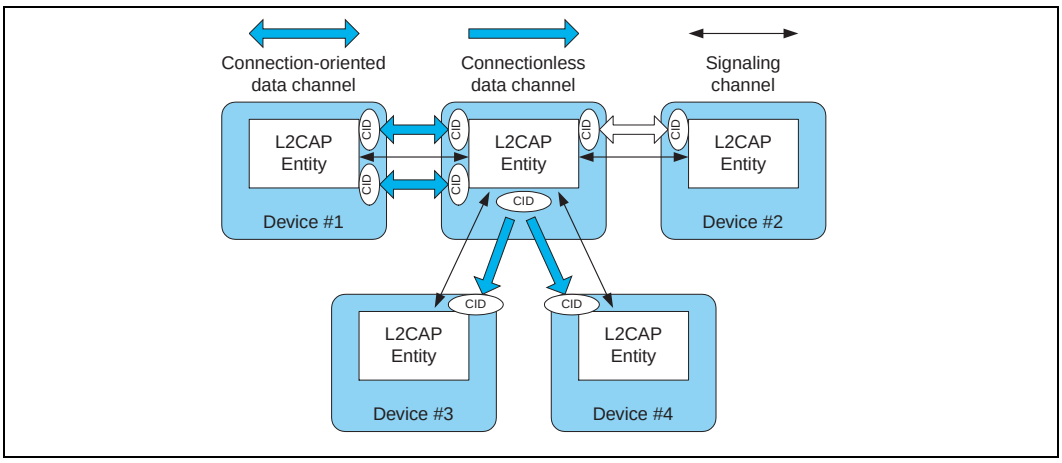

Figure 2.1: Channels between devices

Table 2.2 describes the various channels and their source and destination identifiers. An “allocated” channel is created to represent the local endpoint and should be in the range 0x0040 to 0xFFFF. Section 3 on page 258 describes the state machine associated with each connectionless channel. Section 4.1 on page 272 describes the packet format associated with bi-directional channels and Section 4.2 on page 272 describes the packet format associated with uni-directional channels.

Channel Type	Local CID	Remote CID
Connection-oriented	Dynamically allocated	Dynamically allocated
Connectionless data	Dynamically allocated	0x0002 (fixed)
Signalling	0x0001 (fixed)	0x0001 (fixed)

Table 2.2: Types of Channel Identifiers

2.3 OPERATION BETWEEN LAYERS

L2CAP implementations should follow the general architecture described below. L2CAP implementations must transfer data between higher layer protocols and the lower layer protocol. This document lists a number of services that should be exported by any L2CAP implementation. Each implementation must also support a set of signalling commands for use between L2CAP implementations. L2CAP implementations should also be prepared to accept certain types of events from lower layers and generate events to upper layers. How these events are passed between layers is an implementation-dependent process.

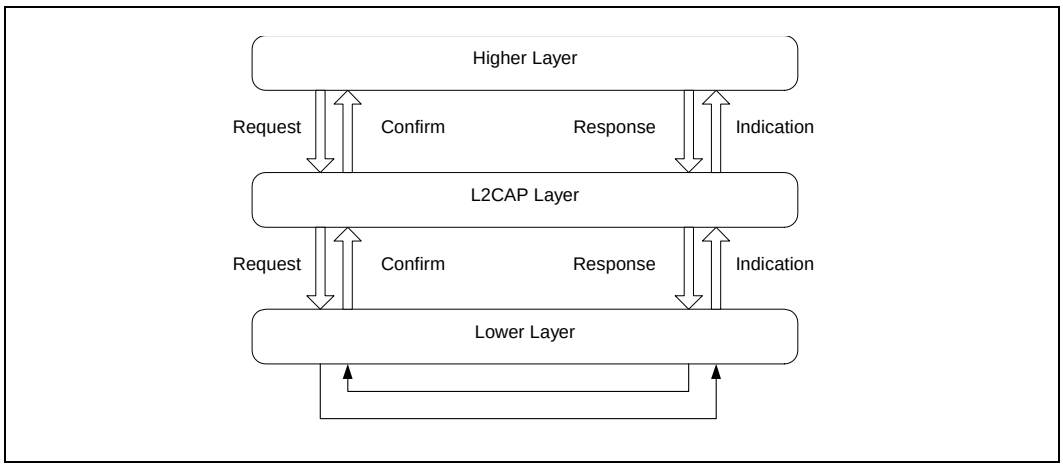

Figure 2.2: L2CAP Architecture

2.4 SEGMENTATION AND REASSEMBLY

Segmentation and reassembly (SAR) operations are used to improve efficiency by supporting a maximum transmission unit (MTU) size larger than the largest Baseband packet. This reduces overhead by spreading the network and transport packets used by higher layer protocols over several Baseband packets. All L2CAP packets may be segmented for transfer over Baseband packets. The protocol does not perform any segmentation and reassembly operations but the packet format supports adaptation to smaller physical frame sizes. An L2CAP implementation exposes the outgoing (i.e., the remote host's receiving) MTU and segments higher layer packets into "chunks" that can be passed to the Link Manager via the Host Controller Interface (HCI), whenever one exists. On the receiving side, an L2CAP implementation receives "chunks" from the HCI and reassembles those chunks into L2CAP packets using information provided through the HCI and from the packet header.

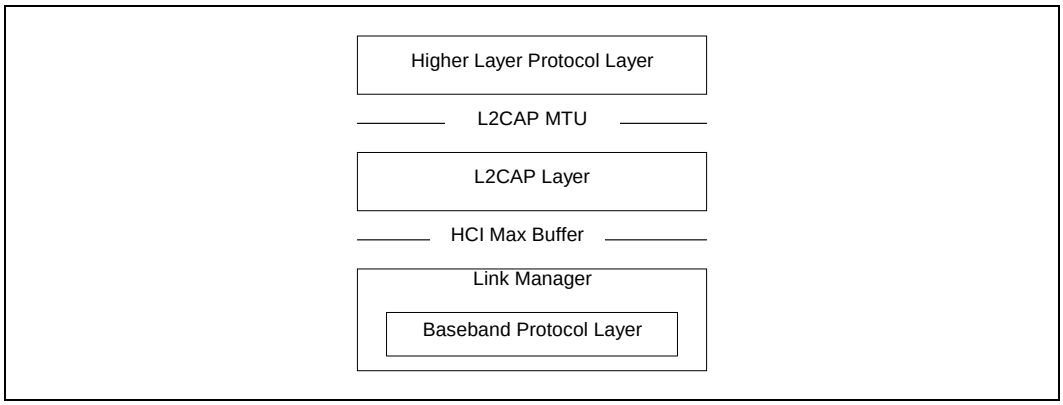

Figure 2.3: L2CAP SAR Variables

Segmentation and Reassembly is implemented using very little overhead in Baseband packets. The two L_CH bits defined in the first byte of Baseband

payload (also called the frame header) are used to signal the start and continuation of L2CAP packets. L_CH shall be “10” for the first segment in an L2CAP packet and “01” for a continuation segment. An example use of SAR is shown in Figure 2.4.

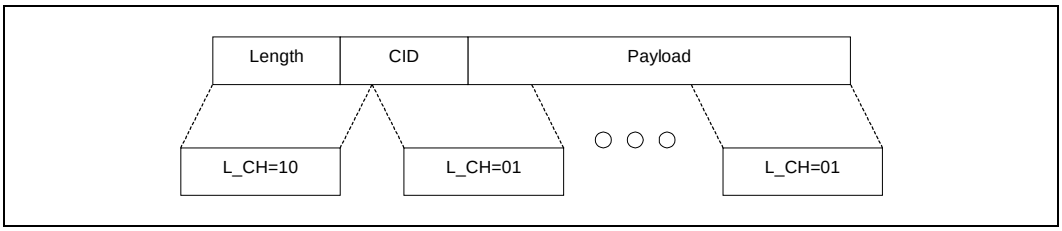

Figure 2.4: L2CAP segmentation

2.4.1 Segmentation Procedures

The L2CAP maximum transmission unit (MTU) will be exported using an implementation spec8ific service interface. It is the responsibility of the higher layer protocol to limit the size of packets sent to the L2CAP layer below the MTU limit. An L2CAP implementation will segment the packet into protocol data units (PDUs) to send to the lower layer. If L2CAP runs directly over the Baseband Protocol, an implementation may segment the packet into Baseband packets for transmission over the air. If L2CAP runs above the host controller interface (typical scenario), an implementation may send block-sized chunks to the host controller where they will be converted into Baseband packets. All L2CAP segments associated with an L2CAP packet must be passed through to the Baseband before any other L2CAP packet destined to the same unit may be sent.

2.4.2 Reassembly Procedures

The Baseband Protocol delivers ACL packets in order and protects the integrity of the data using a 16-bit CRC. The Baseband also supports reliable connections using an automatic repeat request (ARQ) mechanism. As the Baseband controller receives ACL packets, it either signals the L2CAP layer on the arrival of each Baseband packets, or accumulates a number of packets before the receive buffer fills up or a timer expires before signalling the L2CAP layer.

L2CAP implementations must use the length field in the header of L2CAP packets, see Section 4 on page 272, as a consistency check and discard any L2CAP packets that fail to match the length field. If channel reliability is not needed, packets with improper lengths may be silently discarded. For reliable channels, L2CAP implementations must indicate to the upper layer that the channel has become unreliable. Reliable channels are defined by having an infinite flush timeout value as specified in Section 6.2 on page 289.

Figure 2.5 on page 257 illustrates the use of segmentation and reassembly operations to transmit a single higher layer PDU. Note that while there is a one-to-one mapping between a high layer PDU and an L2CAP packet, the segment size used by the segmentation and reassembly routines is left to the implementation and may differ from the sender to the receiver.

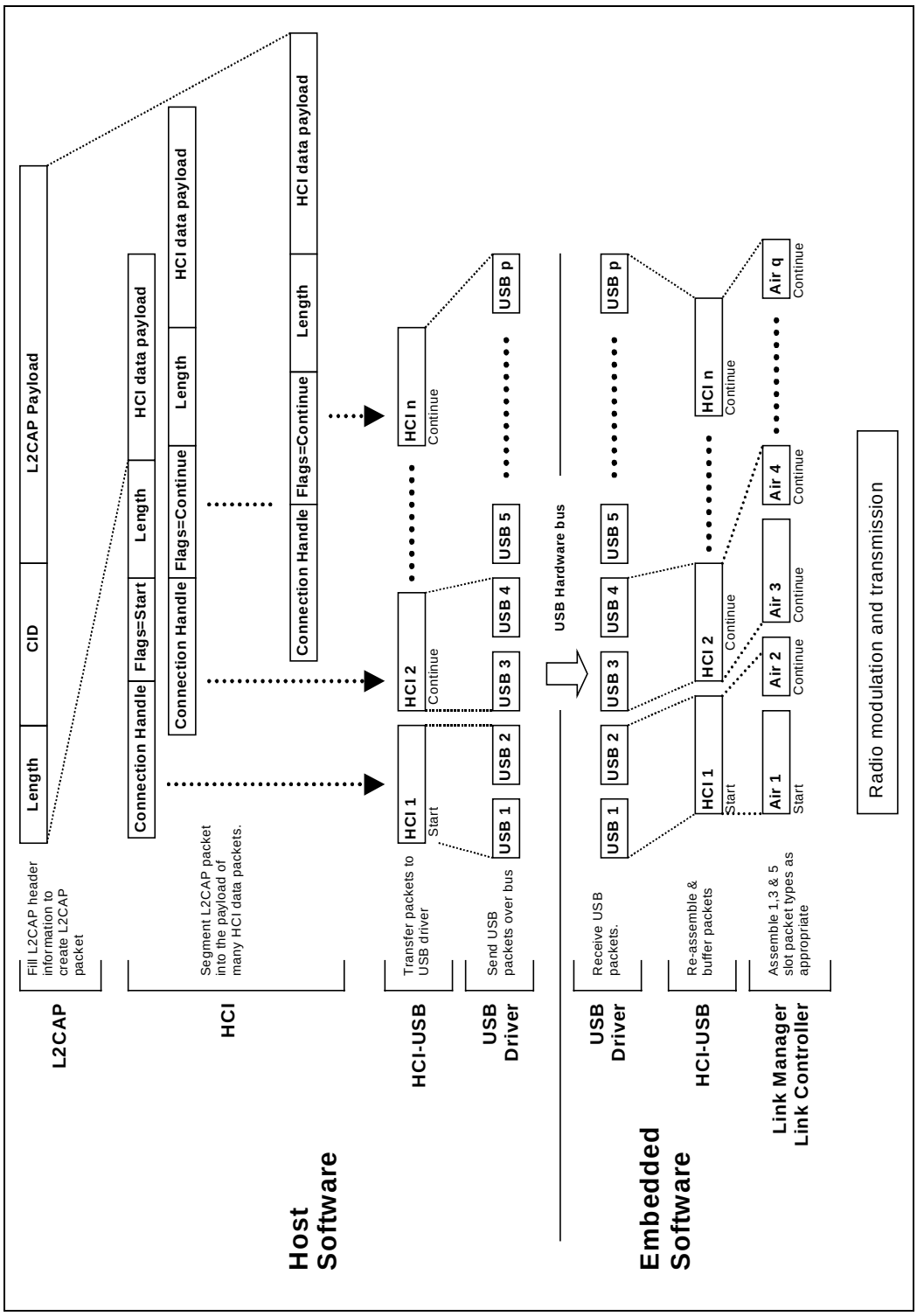

Figure 2.5: Segmentation and Reassembly Services in a unit with an HCI[1]

1. For simplicity, the stripping of any additional HCI and USB specific information fields prior to the creation of the baseband packets (Air_1, Air_2, etc.) is not shown in the figure.

3 STATE MACHINE

This section describes the L2CAP connection-oriented channel state machine. The section defines the states, the events causing state transitions, and the actions to be performed in response to events. This state machine is only pertinent to bi-directional CIDs and is not representative of the signalling channel or the uni-directional channel.

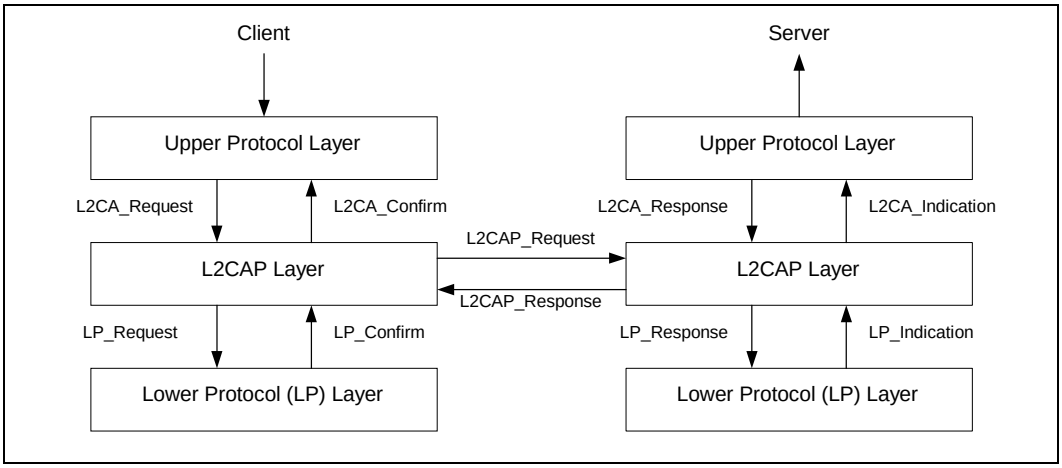

Figure 3.1: L2CAP Layer Interactions

Figure 3.1 illustrates the events and actions performed by an implementation of the L2CAP layer. Client and Server simply represent the initiator of the request and the acceptor of the request respectively. An application-level Client would both initiate and accept requests. The naming convention is as follows. The interface between two layers (vertical interface) uses the prefix of the lower layer offering the service to the higher layer, e.g., L2CA. The interface between two entities of the same layer (horizontal interface) uses the prefix of the protocol (adding a P to the layer identification), e.g., L2CAP. Events coming from above are called Requests (Req) and the corresponding replies are called Confirms (Cfm). Events coming from below are called Indications (Ind) and the corresponding replies are called Responses (Rsp). Responses requiring further processing are called Pending (Pnd). The notation for Confirms and Responses assumes positive replies. Negative replies are denoted by a “Neg” postfix such as L2CAP_ConnectCfmNeg.

While Requests for an action always result in a corresponding Confirmation (for the successful or unsuccessful satisfaction of the action), Indications do not always result into corresponding Responses. The latter is especially true, if the Indications are informative about locally triggered events, e.g., seeing the *LP_QoSViolationInd* in Section 3.1.1 on page 259, or *L2CA_TimeOutInd* in Section 3.2.4 on page 264.

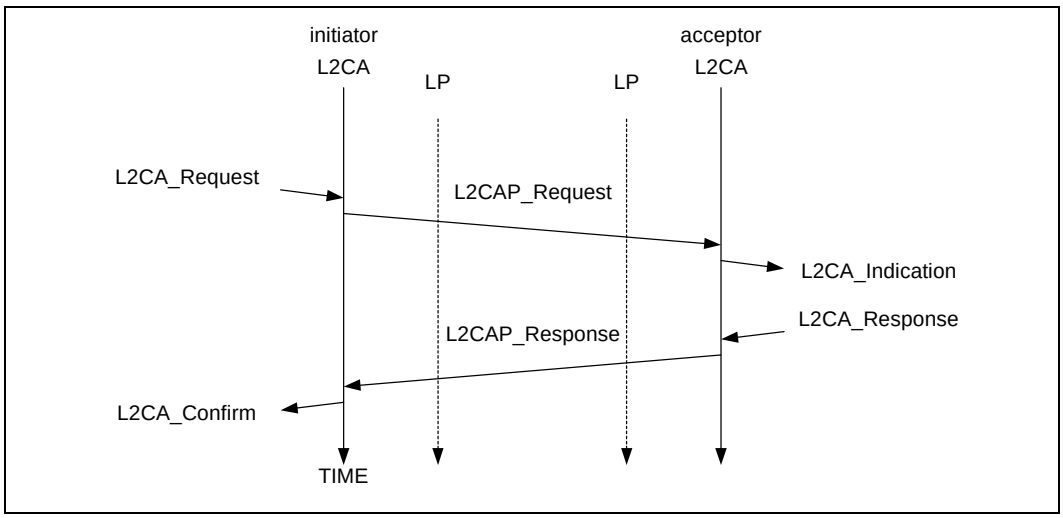

Figure 3.2: MSC of Layer Interactions

Figure 3.2 uses a message sequence chart (MSC) to illustrate the normal sequence of events. The two outer vertical lines represent the L2CA interface on the initiator (the device issuing a request) and the acceptor (the device responding to the initiator's request). Request commands at the L2CA interface result in Requests defined by the protocol. When the protocol communicates the request to the acceptor, the remote L2CA entity presents the upper protocol with an Indication. When the acceptor's upper protocol responds, the response is packaged by the protocol and communicated back the to initiator. The result is passed back to the initiator's upper protocol using a Confirm message.

3.1 EVENTS

Events are all incoming messages to the L2CA layer along with time-outs. Events are partitioned into five categories: Indications and Confirms from lower layers, Requests and Responses from higher layers, data from peers, signal Requests and Responses from peers, and events caused by timer expirations.

3.1.1 Lower-Layer Protocol (LP) to L2CAP events

- *LP_ConnectCfm*

  Confirms the request (see *LP_ConnectReq* in Section 3.2.1) to establish a lower layer (Baseband) connection. This includes passing the authentication challenge if authentication is required to establish the physical link.

- *LP_ConnectCfmNeg*

  Confirms the failure of the request (see *LP_ConnectReq* in Section 3.2.1) to establish a lower layer (Baseband) connection failed. This could be because the device could not be contacted, refused the request, or the LMP authentication challenge failed.

- *LP_ConnectInd*

  Indicates the lower protocol has successfully established connection. In the case of the Baseband, this will be an ACL link. An L2CAP entity may use to information to keep track of what physical links exist.

- *LP_DisconnectInd*

  Indicates the lower protocol (Baseband) has been shut down by LMP commands or a timeout event.

- *LP_QoSCfm*

  Confirms the request (see *LP_QoSReq* in Section 3.2.1) for a given quality of service.

- *LP_QoSCfmNeg*

  Confirms the failure of the request (see *LP_QoSReq* in Section 3.2.1) for a given quality of service.

- *LP_QoSViolationInd*

  Indicates the lower protocol has detected has a violation of the QoS agreement specified in the previous *LP_QoSReq* (see Section 3.2.1).

3.1.2 L2CAP to L2CAP Signalling events

L2CAP to L2CAP signalling events are generated by each L2CAP entity following the exchange of the corresponding L2CAP signalling PDUs, see Section 5. L2CAP signalling PDUs, like any other L2CAP PDUs, are received from a lower layer via a lower protocol indication event. For simplicity of the presentation, we avoid a detailed description of this process, and we assume that signalling events are exchanged directly between the L2CAP peer entities as shown in Figure 3.1 on page 258.

- *L2CAP_ConnectReq*

  A Connection Request packet has been received.

- *L2CAP_ConnectRsp*

  A Connection Response packet has been received with a positive result indicating that the connection has been established.

- *L2CAP_ConnectRspPnd*

  A Connection Response packet has been received indicating the remote endpoint has received the request and is processing it.

- *L2CAP_ConnectRspNeg*

  A Connection Response packet has been received, indicating that the connection could not be established.

- *L2CAP_ConfigReq*

  A Configuration Request packet has been received indicating the remote endpoint wishes to engage in negotiations concerning channel parameters.

- *L2CAP_ConfigRsp*

  A Configuration Response packet has been received indicating the remote endpoint agrees with all the parameters being negotiated.

- *L2CAP_ConfigRspNeg*

  A Configuration Response packet has been received indicating the remote endpoint does not agree to the parameters received in the response packet.

- *L2CAP_DisconnectReq*

  A Disconnection Request packet has been received and the channel must initiate the disconnection process. Following the completion of an L2CAP channel disconnection process, an L2CAP entity should return the corresponding local CID to the pool of “unassigned” CIDs.

- *L2CAP_DisconnectRsp*

  A Disconnection Response packet has been received. Following the receipt of this signal, the receiving L2CAP entity may return the corresponding local CID to the pool of unassigned CIDs. There is no corresponding negative response because the Disconnect Request must succeed.

3.1.3 L2CAP to L2CAP Data events

- *L2CAP_Data*

  A Data packet has been received.

3.1.4 Upper-Layer to L2CAP events

- *L2CA_ConnectReq*

  Request from upper layer for the creation of a channel to a remote device.

- *L2CA_ConnectRsp*

  Response from upper layer to the indication of a connection request from a remote device (see L2CA_ConnectInd in Section 3.2.4).

- *L2CA_ConnectRspNeg*

  Negative response (rejection) from upper layer to the indication of a connection request from a remote device (see *L2CA_ConnectInd* in Section 3.2.4).

- *L2CA_ConfigReq*

  Request from upper layer to (re)configure the channel.

- *L2CA_ConfigRsp*

  Response from upper layer to the indication of a (re)configuration request (see *L2CA_ConfigInd* in Section 3.2.4).

- *L2CA_ConfigRspNeg*

  A negative response from upper layer to the indication of a (re)configuration request (see *L2CA_ConfigInd* in Section 3.2.4).

- *L2CA_DisconnectReq*

  Request from upper layer for the immediate disconnection of a channel.

- *L2CA_DisconnectRsp*

  Response from upper layer to the indication of a disconnection request (see *L2CA_DisconnectInd* in Section 3.2.4). There is no corresponding negative response, the disconnect indication must always be accepted.

- *L2CA_DataRead*

  Request from upper layer for the transfer of received data from L2CAP entity to upper layer.

- *L2CA_DataWrite*

  Request from upper layer for the transfer of data from the upper layer to L2CAP entity for transmission over an open channel.

3.1.5 Timer events

- *RTX*

  The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signalling requests. This timer is started when a signalling request (see Section 5 on page 275) is sent to the remote device. This timer is disabled when the response is received. If the initial timer expires, a duplicate Request message may be sent or the channel identified in the request may be disconnected. If a duplicate Request message is sent, the RTX timeout value must be reset to a new value at least double the previous value.

  Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before disconnecting the channel. The decision should be based on the flush timeout of the signalling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level. For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level.

  The value of this timer is implementation dependent but the minimum initial value is 1 second and the maximum initial value is 60 seconds. One RTX timer MUST exist for each outstanding signalling request, including each Echo Request. The timer disappears on the final expiration, when the response is received, or the physical link is lost. The maximum elapsed time between the initial start of this timer and the initiation of channel disconnection (if no response is received) is 60 seconds.

- *ERTX*

  The Extended Response Timeout eXpired (ERTX) timer is used in place of the RTX timer when it is suspected the remote endpoint is performing additional processing of a request signal. This timer is started when the remote endpoint responds that a request is pending, e.g., when an *L2CAP_ConnectRspPnd* event is received. This timer is disabled when the formal response is received or the physical link is lost. If the initial timer expires, a duplicate Request may be sent or the channel may be discon-

nected. If a duplicate Request is sent, the particular ERTX timer disappears, replaced by a new RTX timer and the whole timing procedure restarts as described previously for the RTX timer.

The value of this timer is implementation dependent but the minimum initial value is 60 seconds and the maximum initial value is 300 seconds. Similar to RTX, there MUST be at least one ERTX timer for each outstanding request that received a Pending response. There should be at most one (RTX or ERTX) associated with each outstanding request. The maximum elapsed time between the initial start of this timer and the initiation of channel disconnection (if no response is received) is 300 seconds.

3.2 ACTIONS

Actions are partitioned into five categories: Confirms and Indications to higher layers, Request and Responses to lower layers, Requests and Responses to peers, data transmission to peers, and setting timers.

3.2.1 L2CAP to Lower Layer actions

- *LP_ConnectReq*

  L2CAP requests the lower protocol to create a connection. If a physical link to the remote device does not exist, this message must be sent to the lower protocol to establish the physical connection. Since no more than a single ACL link between two devices is assumed, see Section 1.2 on page 252, additional L2CAP channels between these two devices must share the same baseband ACL link.
  Following the processing of the request, the lower layer returns with an *LP_ConnectCfm* or an *LP_ConnectCfmNeg* to indicate whether the request has been satisfied or not, respectively.

- *LP_QoSReq*

  L2CAP requests the lower protocol to accommodate a particular QoS parameter set. Following the processing of the request, the lower layer returns with an *LP_QoSCfm* or an *LP_QoSCfmNeg* to indicate whether the request has been satisfied or not, respectively

- *LP_ConnectRsp*

  A positive response accepting the previous connection indication request (see *LP_ConnectInd* in Section 3.1.1).

- *LP_ConnectRspNeg*

  A negative response denying the previous connection indication request (see *LP_ConnectInd* in Section 3.1.1).

3.2.2 L2CAP to L2CAP Signalling actions

This section contains the same names identified in Section 3.1.2 except the actions refer to the transmission, rather than reception, of these messages.

3.2.3 L2CAP to L2CAP Data actions

This section is the counterpart of Section 3.1.3. Data transmission is the action performed here.

3.2.4 L2CAP to Upper Layer actions

- *L2CA_ConnectInd*

  Indicates a Connection Request has been received from a remote device (see *L2CA_ConnectReq* in Section 3.1.4).

- *L2CA_ConnectCfm*

  Confirms that a Connection Request has been accepted (see (*L2CAP_ConnectReq* in Section 3.1.4) following the receipt of a Connection message from the remote device.

- *L2CA_ConnectCfmNeg*

  Negative confirmation (failure) of a Connection Request (see *L2CA_ConnectReq* in Section 3.1.4). An RTX timer expiration (see Section 3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Connect Request can substitute for a negative Connect Response and result in this action.

- *L2CA_ConnectPnd*

  Confirms that a Connection Response (pending) has been received from the remote device.

- *L2CA_ConfigInd*

  Indicates a Configuration Request has been received from a remote device.

- *L2CA_ConfigCfm*

  Confirms that a Configuration Request has been accepted (see *L2CA_ConfigReq* in Section 3.1.4) following the receipt of a Configuration Response from the remote device.

- *L2CA_ConfigCfmNeg*

  Negative confirmation (failure) of a Configuration Request (see *L2CA_ConfigReq* in Section 3.1.4). An RTX timer expiration (see Section 3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Connect Request can substitute for a negative Connect Response and result in this action.

- *L2CA_DisconnectInd*

  Indicates a Disconnection Request has been received from a remote device or the remote device has been disconnected because it has failed to respond to a signalling request. See Section 3.1.5

- *L2CA_DisconnectCfm*

  Confirms that a Disconnect Request has been processed by the remote device (see *L2CA_DisconnectReq* in Section 3.1.4) following the receipt of a Disconnection Response from the remote device. An RTX timer expiration

(see Section 3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Disconnect Request can substitute for a Disconnect Response and result in this action. Upon receiving this event the upper layer knows the L2CAP channel has been terminated. There is no corresponding negative confirm.

- *L2CA_TimeOutInd*

  Indicates that a RTX or ERTX timer has expired. This indication will occur an implementation dependant number of times before the L2CAP implementation will give up and send a L2CA_DisconnectInd.

- *L2CA_QoSViolationInd*

  Indicates that the quality of service agreement has been violated.

3.3 CHANNEL OPERATIONAL STATES

- *CLOSED*

  In this state, there is no channel associated with this CID. This is the only state when a link level connection (Baseband*)* may not exist. Link disconnection forces all other states into the CLOSED state.

- *W4_L2CAP_CONNECT_RSP*

  In this state, the CID represents a local end-point and an L2CAP_ConnectReq message has been sent referencing this endpoint and it is now waiting for the corresponding L2CAP_ConnectRsp message.

- *W4_L2CA_CONNECT_RSP*

  In this state, the remote end-point exists and an *L2CAP_ConnectReq* has been received by the local L2CAP entity. An L2CA_ConnectInd has been sent to the upper layer and the part of the local L2CAP entity processing the received *L2CAP_ConnectReq* waits for the corresponding response. The response may require a security check to be performed.

- *CONFIG*

  In this state, the connection has been established but both sides are still negotiating the channel parameters. The Configuration state may also be entered when the channel parameters are being renegotiated. Prior to entering the CONFIG state, all outgoing data traffic should be suspended since the traffic parameters of the data traffic are to be renegotiated. Incoming data traffic must be accepted until the remote channel endpoint has entered the CONFIG state.

  In the CONFIG state, both sides must issue L2CAP_ConfigReq messages – if only defaults are being used, a null message should be sent, see Section 5.4 on page 280. If a large amount of parameters need to be negotiated, multiple messages may be sent to avoid any MTU limitations and negotiate incrementally – see Section 6 on page 288 for more details.

  Moving from the CONFIG state to the OPEN state requires both sides to be ready. An L2CAP entity is ready when it has received a positive response to its final request and it has positively responded to the final request from the remote device.

- *OPEN*

  In this state, the connection has been established and configured, and data flow may proceed.

- *W4_L2CAP_DISCONNECT_RSP*

  In this state, the connection is shutting down and an L2CAP_DisconnectReq message has been sent. This state is now waiting for the corresponding response.

- *W4_L2CA_DISCONNECT_RSP*

  In this state, the connection on the remote endpoint is shutting down and an L2CAP_DisconnectReq message has been received. An L2CA_DisconnectInd has been sent to the upper layer to notify the owner of the CID that the remote endpoint is being closed. This state is now waiting for the corresponding response from the upper layer before responding to the remote endpoint.

3.4 MAPPING EVENTS TO ACTIONS

Table 3.1 defines the actions taken in response to events that occur in a particular state. Events that are not listed in the table, nor have actions marked N/A, are assumed to be errors and silently discarded.

Data input and output events are only defined for the Open and Configuration states. Data may not be received during the initial Configuration state, but may be received when the Configuration state is re-entered due to a reconfiguration process. Data received during any other state should be silently discarded.

Event	Current State	Action	New State
LP_ConnectCfm	CLOSED	Flag physical link as up and initiate the L2CAP connection.	CLOSED
LP_ConnectCfmNeg	CLOSED	Flag physical link as down and fail any outstanding service connection requests by sending an L2CA_ConnectCfmNeg message to the upper layer.	CLOSED
LP_ConnectInd	CLOSED	Flag link as up.	CLOSED
LP_DisconnectInd	CLOSED	Flag link as down.	CLOSED
LP_DisconnectInd	Any except CLOSED	Send upper layer L2CA_DisconnectInd message.	CLOSED

Table 3.1: L2CAP Channel State Machine

Event	Current State	Action	New State
LP_QoSViolationInd	Any but OPEN	Discard	N/C
LP_QoSViolationInd	OPEN	Send upper layer L2CA_QoSViolationInd message. If service level is guaranteed, terminate the channel.	OPEN or W4_L2CA_DISCON_RSP
L2CAP_ConnectReq	CLOSED. (CID dynamically allocated from free pool.)	Send upper layer L2CA_ConnectInd. Optionally: Send peer L2CAP_ConnectRspPnd	W4_L2CA_CONNECT_RSP
L2CAP_ConnectRsp	W4_L2CAP_CONECT_RSP	Send upper layer L2CA_ConnectCfm message. Disable RTX timer.	CONFIG
L2CAP_ConnectRspPnd	W4_L2CAP_CONECT_RSP	Send upper layer L2CA_ConnectPnd message. Disable RTX timer and start ERTX timer.	N/
L2CAP_ConnectRspNeg	W4_L2CAP_CONECT_RSP	Send upper layer L2CA_ConnectCfmNeg message. Return CID to free pool. Disable RTX/ERTX timers.	CLOSED
L2CAP_ConfigReq	CLOSED	Send peer L2CAP_ConfigRspNeg message.	N/C
L2CAP_ConfigReq	CONFIG	Send upper layer L2CA_ConfigInd message.	N/C
L2CAP_ConfigReq	OPEN	Suspend data transmission at a convenient point. Send upper layer L2CA_ConfigInd message.	CONFIG
L2CAP_ConfigRsp	CONFIG	Send upper layer L2CA_ConfigCfm message. Disable RTX timer. If an L2CAP_ConfigReq message has been received and positively responded to, then enter OPEN state, otherwise remain in CONFIG state.	N/C or OPEN
L2CAP_ConfigRspNeg	CONFIG	Send upper layer L2CA_ConfigCfmNeg message. Disable RTX timer.	N/C

Table 3.1: L2CAP Channel State Machine

Event	Current State	Action	New State
L2CAP_Disconnect Req	CLOSED	Send peer L2CAP_DisconnectRsp message.	N/C
L2CAP_Disconnect Req	!CLOSED	Send upper layer L2CA_ DisconnectInd message.	W4_L2CA_ DISCON_RSP
L2CAP_Disconnect Rsp	W4_L2CAP_DISC ON_RSP	Send upper layer L2CA_DisconnectCfm mes-sage. Disable RTX timer.	CLOSED
L2CAP_Data	OPEN or CONFIG	If complete L2CAP packet received, send upper layer L2CA_Read confirm.	N/C
L2CA_ConnectReq	CLOSED (CID dynamically allocated from free pool)	Send peer LP2CAP_ConnectReq message. Start RTX timer.	W4_L2CAP_ CONECT_RSP
L2CA_ConnectRsp	W4_L2CA_CONN ECT_RSP	Send peer L2CAP_ ConnectRsp message. Optionally: Send L2CAP_ConfigReq message.	CONFIG
L2CA_ConnectRsp Neg	W4_L2CA_CONN ECT_RSP	Send peer L2CAP_ConnectRspNeg message. Return CID to free pool.	CLOSED
L2CA_ConfigReq	CLOSED	Send upper layer L2CA_ ConfigCfmNeg message.	N/C
L2CA_ConfigReq	CONFIG	Send peer L2CAP_ConfigReq message. Start RTX timer.	N/C
L2CA_ConfigReq	OPEN	Suspend data transmission at a convenient point. Send peer L2CAP_ConfigReq message. Start RTX timer.	CONFIG
L2CA_ConfigRsp	CONFIG	Send peer L2CAP_ConfigRsp message. If all outstanding L2CAP_ConfigReq mes-sages have received posi-tive responses then move in OPEN state. Otherwise, remain in CONFIG state.	N/C or OPEN
L2CA_ConfigRspNeg	CONFIG	Send peer L2CAP_ ConfigRspNeg message.	N/C

Table 3.1: L2CAP Channel State Machine

Event	Current State	Action	New State
L2CA_Disconnect Req	OPEN	Send peer L2CAP_DisconnectReq message. Start RTX timer.	W4_L2CAP_ DISCON_RSP
L2CA_DisconnectRsp	W4_L2CA_DISC ON_RSP	Send peer L2CAP_DisconnectRsp message. Return CID to free pool.	CLOSED
L2CA_DataRead	OPEN	If payload complete, trans- fer payload to InBuffer.	OPEN
L2CA_DataWrite	OPEN	Send peer L2CAP_Data message.	OPEN
Timer_RTX	Any	Send upper layer L2CA_TimeOutInd message. Return CID to free pool.	CLOSED
Timer_ERTX	Any	Send upper layer L2CA_TimeOutInd message. Return CID to free pool.	CLOSED

Table 3.1: L2CAP Channel State Machine

Figure 3.3 illustrates a simplified state machine and typical transition path taken by an initiator and acceptor. The state machine shows what events cause state transitions and what actions are also taken while the transitions occur. Not all the events listed in Table 3.1 are included in the simplified State Machine to avoid cluttering the figure.

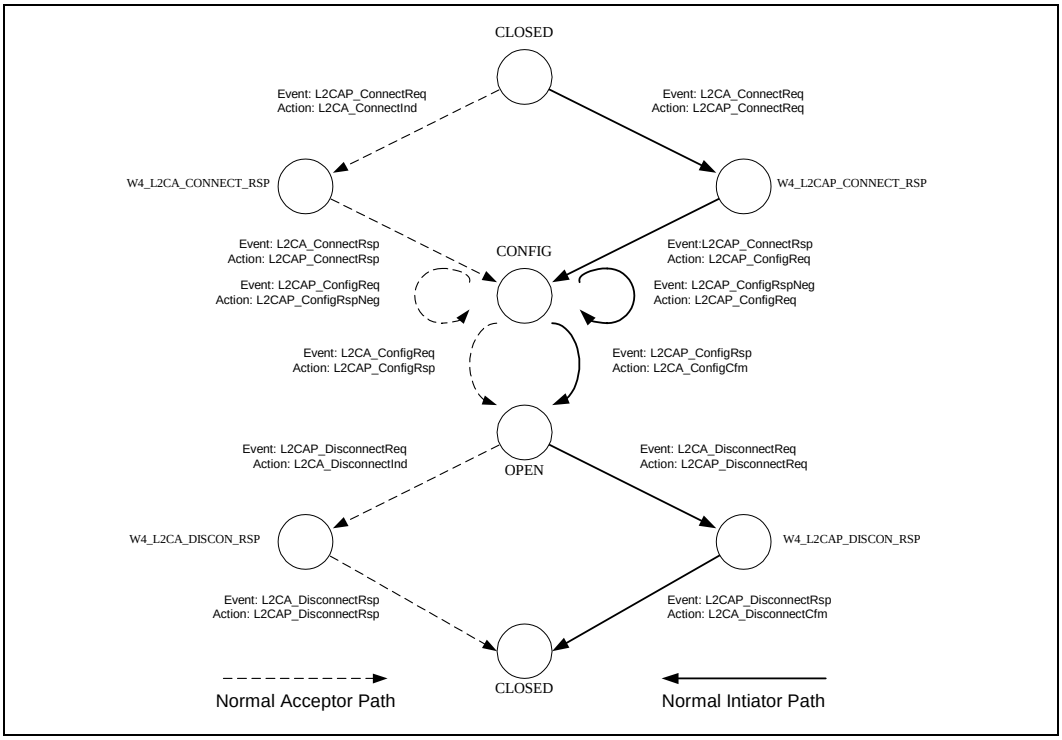

Figure 3.3: State Machine Example

Figure 3.4 presents another illustration of the events and actions based around the messages sequences being communicated between two devices. In this example, the initiator is creating the first L2CAP channel between two devices. Both sides start in the CLOSED state. After receiving the request from the upper layer, the entity requests the lower layer to establish a physical link. If no physical link exists, LMP commands are used to create the physical link between the devices. Once the physical link is established, L2CAP signals may be sent over it.

Figure 3.4 is an example and not all setup sequences will be identical to the one illustrated below.

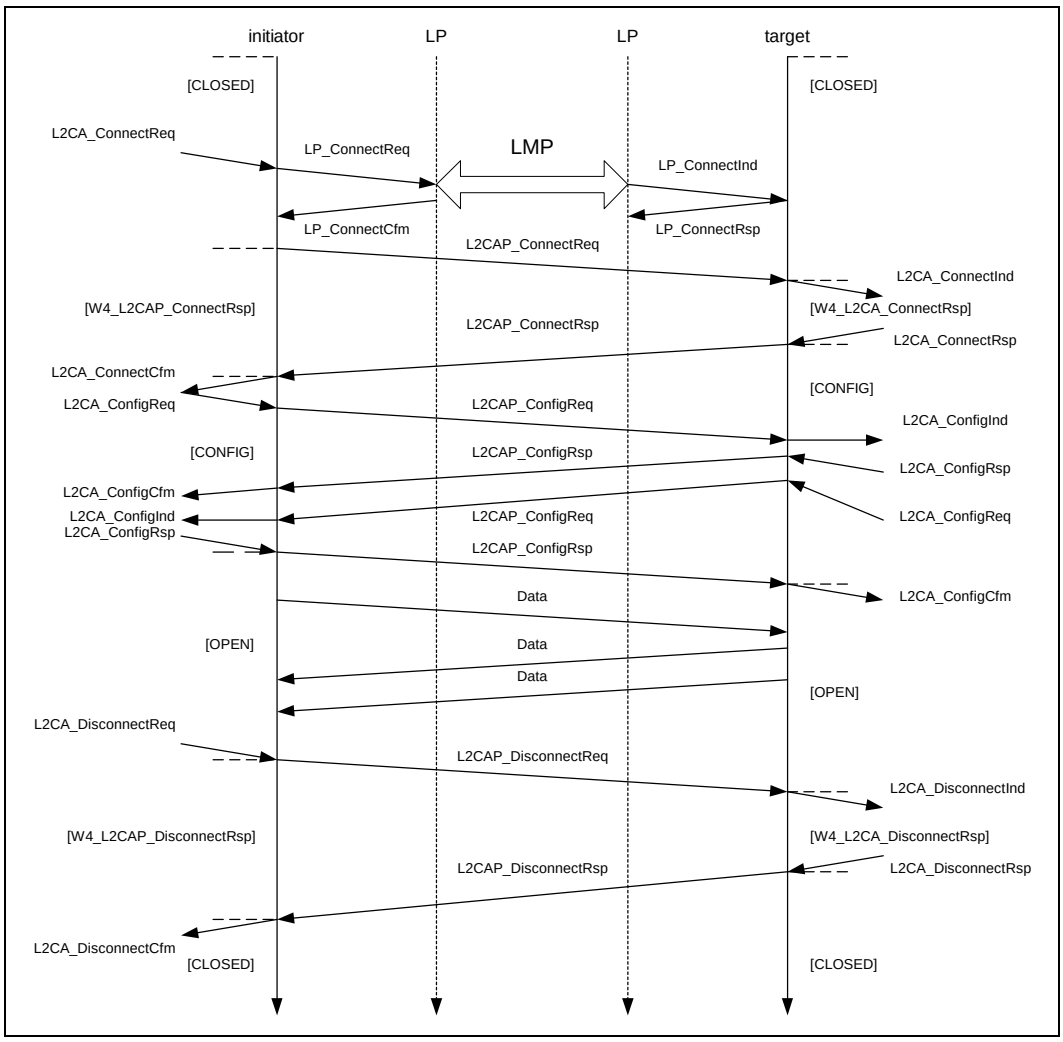

Figure 3.4: Message Sequence Chart of Basic Operation

4 DATA PACKET FORMAT

L2CAP is packet-based but follows a communication model based on *channels*. A channel represents a data flow between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless.

4.1 CONNECTION-ORIENTED CHANNEL

Figure 4.1 illustrates the format of the L2CAP packet (also referred to as the L2CAP PDU) within a connection-oriented channel.

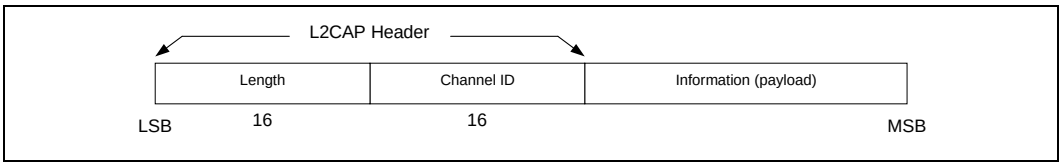

Figure 4.1: L2CAP Packet (field sizes in bits)

The fields shown are:

- *Length: 2 octets (16 bits)*

Length indicates the size of information payload in bytes, excluding the length of the L2CAP header. The length of an information payload can be up to 65535 bytes. The Length field serves as a simple integrity check of the reassembled L2CAP packet on the receiving end.

- *Channel ID: 2 octets*

  The channel ID identifies the destination channel endpoint of the packet. The scope of the channel ID is relative to the device the packet is being sent to.

- *Information: 0 to 65535 octets*

  This contains the payload received from the upper layer protocol (outgoing packet), or delivered to the upper layer protocol (incoming packet). The minimum supported MTU for connection-oriented packets (MTU_{cno}) is negotiated during channel configuration (see Section 6.1 on page 288). The minimum supported MTU for the signalling packet (MTU_{sig}) is 48 bytes (see Section 5 on page 275).

4.2 CONNECTIONLESS DATA CHANNEL

In addition to connection-oriented channels, L2CAP also exports the concept of a group-oriented channel. Data sent to the “group” channel is sent to all members of the group in a best effort manner. Groups have no quality of service associated with them. Group channels are unreliable; L2CAP makes no guarantee that data sent to the group successfully reaches all members of the group. If reliable group transmission is required, it must be implemented at a higher layer.

Transmissions to a group must be non-exclusively sent to all members of that group. The local device cannot be a member of the group and higher layer protocols are expected to loopback any data traffic being sent to the local device. Non-exclusive implies non-group members may receive group transmissions and higher level (or link level) encryption can be used to support private communication.

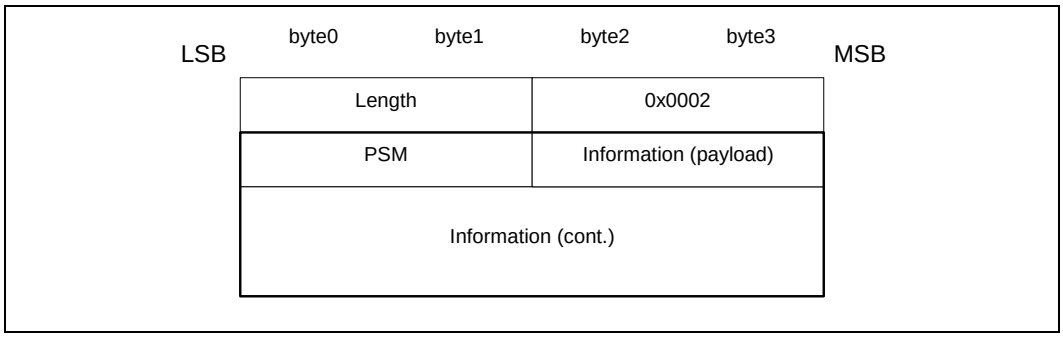

Figure 4.2: Connectionless Packet

The fields shown are:

- *Length: 2 octets*

  Length indicates the size of information payload plus the PSM field in bytes, excluding the length of the L2CAP header.

- *Connectionless Traffic CID: 2 octets*

  Channel ID (0x0002) reserved for connectionless traffic.

- *Protocol/Service Multiplexer (PSM): 2 octets (minimum)*

  The PSM field is based on the ISO 3309 extension mechanism for address fields. All content of the PSM field, referred to as the PSM value, must be ODD, that is, the least significant bit of the least significant octet must be “1”. Also, all PSM values must be assigned such that the least significant bit of the most significant octet equals “0”. This allows the PSM field to be extended beyond 16 bits. The PSM value definitions are specific to L2CAP and assigned by the Bluetooth SIG. For more information on the PSM field see Section 5.2 on page 278.

- *Information: 0 to 65535 octets*

  The payload information to be distributed to all members of the group. Implementations must support a minimum connectionless MTU (MTU_{cnl}) of 670 octets, unless explicitly agreed upon otherwise, e.g., for single operation devices that are built to comply to a specific Bluetooth profile that dictates the use of a specific MTU for connectionless traffic that is less than MTU_{cnl}.

The L2CAP group service interface provides basic group management mechanisms including creating a group, adding members to a group, and removing

members from a group. There are no pre-defined groups such as “all radios in range”.

An entire L2CAP packet sent to a group must be processed before sending any connection-oriented packets to any member of the current piconet and all previous connection-oriented packets must have been submitted to the HCI. The reason for this is the chance messages from the same host overlap because one message is being sent to the receiver and another may be sent to the group and these cannot be demultiplexed correctly. This may have the unfortunate effect of serialising operations when groups are used.

5 SIGNALLING

This section describes the signalling commands passed between two L2CAP entities on remote devices. All signalling commands are sent to CID 0x0001. The L2CAP implementation must be able to determine the Bluetooth address (BD_ADDR) of the device that sent the commands. Figure 5.1 illustrates the general format of all L2CAP packets containing signalling commands. Multiple commands may be sent in a single (L2CAP) packet and packets are sent to CID 0x0001. The minimum MTU for this signalling CID (MTU_{sig}) is 48 bytes.

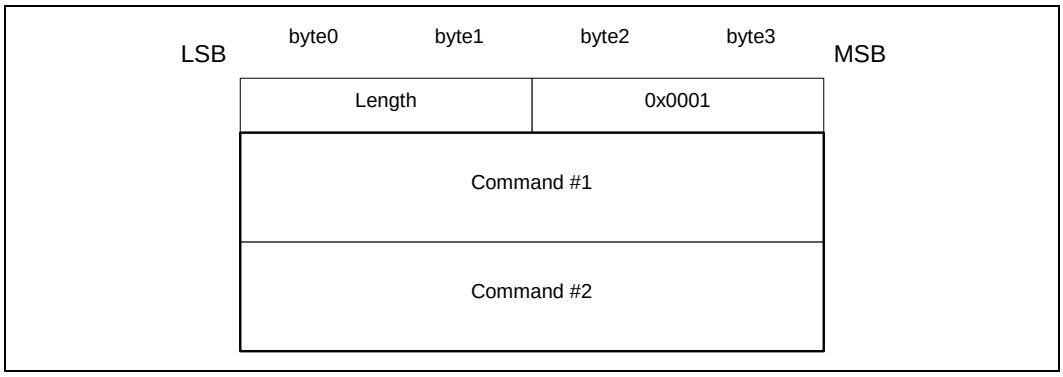

Figure 5.1: Signalling Command Packet Format

Figure 5.2 displays the general format of all signalling commands.

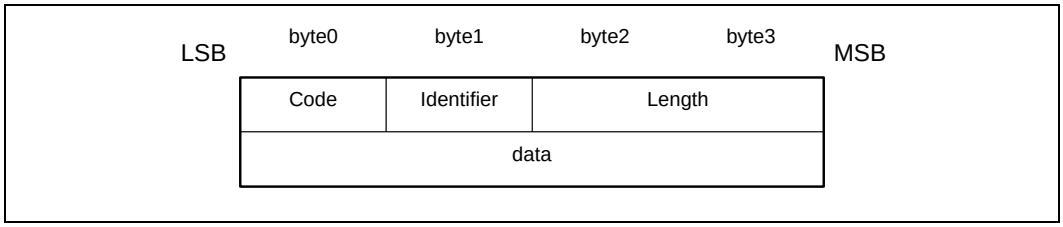

Figure 5.2: Command format

The fields shown are:

- *Code: 1 octet*

  The Code field is one octet long and identifies the type of command. When a packet is received with an unknown Code field, a Command Reject packet (defined in Section 5.1 on page 276) is sent in response.

  Up-to-date values of assigned Codes are specified in the latest Bluetooth “Assigned Numbers” document (page 993). Table 5.1 on page 276 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

Code	Description
0x00	RESERVED
0x01	Command reject
0x02	Connection request
0x03	Connection response
0x04	Configure request
0x05	Configure response
0x06	Disconnection request
0x07	Disconnection response
0x08	Echo request
0x09	Echo response
0x0A	Information request
0x0B	Information response

Table 5.1: Signalling Command Codes

- *Identifier: 1 octet*

  The Identifier field is one octet long and helps matching a request with the reply. The requesting device sets this field and the responding device uses the same value in its response. A different Identifier value must be used for each original command sent for which no response has been received. On the expiration of a RTX or ERTX timer, the same identifier should be used if a duplicate Request is re-sent as stated in Section 3.1.5 on page 262. A device receiving a duplicate request should reply with a duplicate response. A command response with an invalid identifier is silently discarded.

- *Length: 2 octets*

  The Length field is two octets long and indicates the size in octets of the data field of the command only, i.e., it does not cover the Code, Identifier, and Length fields.

- *Data: 0 or more octets*

  The Data field is variable in length and discovered using the Length field. The Code field determines the format of the Data field.

5.1 COMMAND REJECT (CODE 0x01)

A Command Reject packet is sent in response to a command packet with an unknown command code or when sending the corresponding Response is inappropriate. Figure 5.3 displays the format of the packet. The Identifier should match the Identifier of the packet containing the unidentified code field. Implementations must always send these packets in response to unidentified signalling packets.

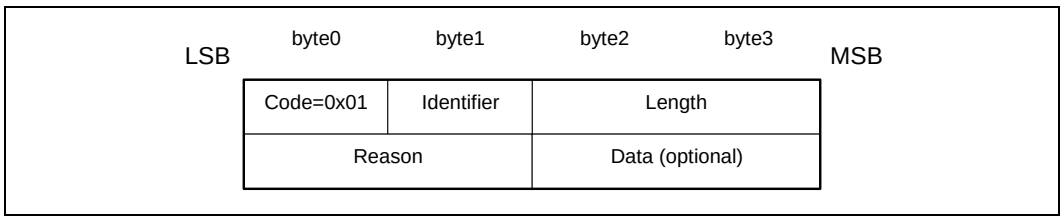

Figure 5.3: Command Reject Packet

- *Length = 0x0002 or more octets*
- *Reason: 2 octets*

  The Reason field describes why the Request packet was rejected.

Reason value	Description
0x0000	Command not understood
0x0001	Signalling MTU exceeded
0x0002	Invalid CID in request
Other	Reserved

Table 5.2: Reason Code Descriptions

- *Data: 0 or more octets*

  The length and content of the Data field depends on the Reason code. If the Reason code is 0x0000, “Command not understood”, no Data field is used. If the Reason code is 0x0001, “Signalling MTU Exceeded”, the 2-octet Data field represents the maximum signalling MTU the sender of this packet can accept.

  If a command refers to an invalid channel then the Reason code 0x0002 will be returned. Typically a channel is invalid because it does not exist. A 4-octet data field on the command reject will contain the local (first) and remote (second) channel endpoints (relative to the sender of the Command Reject) of the disputed channel. The latter endpoints are obtained from the corresponding rejected command. If the rejected command contains only one of the channel endpoints, the other one is replaced by the null CID 0x0000.

Reason value	Data Length	Data value
0x0000	0 octets	N/A
0x0001	2 octets	Actual MTU
0x0002	4 octets	Requested CID

Table 5.3: Reason Data values

5.2 CONNECTION REQUEST (CODE 0x02)

Connection request packets are sent to create a channel between two devices. The channel connection must be established before configuration may begin. Figure 5.4 illustrates a Connection Request packet.

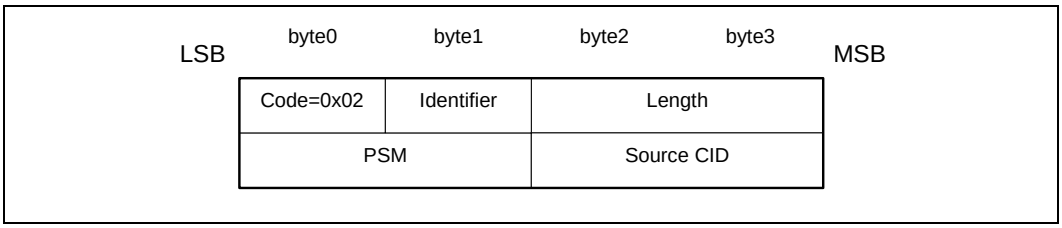

Figure 5.4: Connection Request Packet

- *Length = 0x0004 or more octets*
- *Protocol/Service Multiplexor (PSM): 2 octets (minimum)*

  The PSM field is two octets (minimum) in length. The structure of the PSM field is based on the ISO 3309 extension mechanism for address fields. All PSM values must be ODD, that is, the least significant bit of the least significant octet must be “1”. Also, all PSM values must be assigned such that the least significant bit of the most significant octet equals “0”. This allows the PSM field to be extended beyond 16 bits. PSM values are separated into two ranges. Values in the first range are assigned by the Bluetooth SIG and indicate protocols. The second range of values are dynamically allocated and used in conjunction with the Service Discovery Protocol (SDP). The dynamically assigned values may be used to support multiple implementations of a particular protocol, e.g., RFCOMM, residing on top of L2CAP or for prototyping an experimental protocol.

PSM value	Description
0x0001	Service Discovery Protocol
0x0003	RFCOMM
0x0005	Telephony Control Protocol
<0x1000	RESERVED
[0x1001-0xFFFF]	DYNAMICALLY ASSIGNED

Table 5.4: Defined PSM Values

- *Source CID (SCID): 2 octets*

  The source local CID is two octets in length and represents a channel endpoint on the device sending the request. Once the channel has been configured, data packets flowing from the sender of the request must be send to this CID. In this section, the Source CID represents the channel endpoint on the device sending the request and receiving the response, while the Desti-

nation CID represents the channel endpoint on the device receiving the request and sending the response.

5.3 CONNECTION RESPONSE (CODE 0x03)

When a unit receives a Connection Request packet, it must send a Connection Response packet. The format of the connection response packet is shown in Figure 5.5.

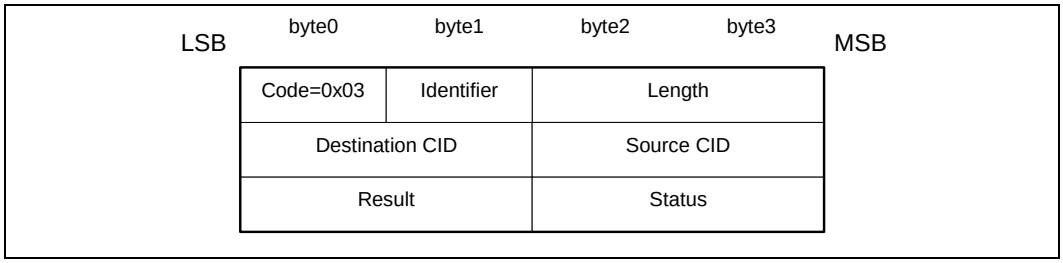

Figure 5.5: Connection Response Packet

- *Length = 0x0008 octets*
- *Destination Channel Identifier (DCID): 2 octets*

  The field contains the channel end-point on the device sending this Response packet.
- *Source Channel Identifier (SCID): 2 octets*

  The field contains the channel end-point on the device receiving this Response packet.
- *Result: 2 octets*

  The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed. A logical channel is established on the receipt of a successful result. Table 5.5 defines values for this field. If the result field is not zero, the DCID and SCID fields should be ignored.

Value	Description
0x0000	Connection successful.
0x0001	Connection pending
0x0002	Connection refused – PSM not supported.
0x0003	Connection refused – security block.
0x0004	Connection refused – no resources available.
Other	Reserved.

Table 5.5: Result values

- *Status: 2 octets*

  Only defined for Result = Pending. Indicates the status of the connection.

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorisation pending
Other	Reserved

Table 5.6: Status values

5.4 CONFIGURATION REQUEST (CODE 0x04)

Configuration Request packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to re-negotiate this contract whenever appropriate. During a re-negotiation session, all data traffic on the channel should be suspended pending the outcome of the negotiation. Each configuration parameter in a Configuration Request is related exclusively either with the outgoing or the incoming data traffic but not both of them. In Section 6 on page 288, the various configuration parameters and their relation to the outgoing or incoming data traffic are presented.

The sender of a successful Connection Response packet, see Section 5.5 on page 282, must also send immediately a Configuration Request packet. If an L2CAP entity receives a Configuration Request while it is waiting for a response it must not block sending the Configuration Response, otherwise the configuration process may deadlock.

If no parameters need to be negotiated, no options need to be inserted and the C-bit should be cleared. L2CAP entities in remote devices MUST negotiate all parameters defined in this document whenever the default values are not acceptable. Any missing configuration parameters are assumed to have their most recently (mutually) explicitly or implicitly accepted values. Event if all default values are acceptable, a Configuration Request packet with no options MUST be sent. Implicitly accepted values are any default values for the configuration parameters specified in this document that have not been explicitly negotiated for the specific channel under configuration.

Each configuration parameter is one-directional and relative to the direction implied by the sender of a Configuration Request. If a device needs to establish the value of a configuration parameter in the opposite direction than the one implied by a Configuration Request, a new Configuration Request with the desired value of the configuration parameter in it needs to be sent in the direction opposite the one used for the original Connection Request.

The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is left to the implementation but it shall not last more than 120 seconds.

Figure 5.6 defines the format of the Configuration Request packet.

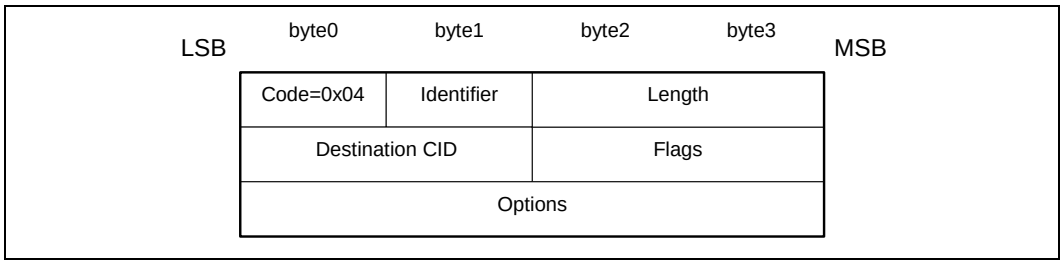

Figure 5.6: Configuration Request Packet

- *Length = 0x0004 or more octets*
- *Destination CID (DCID) : 2 octets*
- The field contains the channel end-point on the device sending this Request packet.
- *Flags: 2 octets*

  Figure 5.7 display the two-octet Flags field. Note the most significant bit is shown on the left.

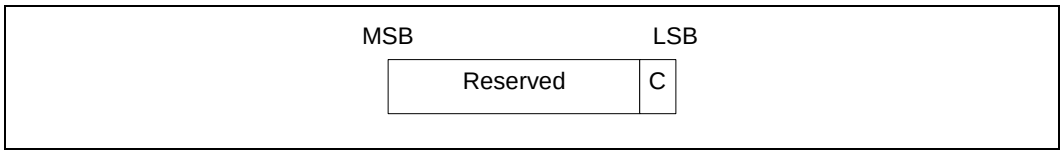

Figure 5.7: Configuration Request Flags field format

C – more configuration requests will follow when set to 1. This flag indicates that the remote device should not enter OPEN state after agreeing to these parameters because more parameter negotiations are being sent. Segmenting the Configuration Request packet is necessary if the parameters exceed the MTU_{sig}.

Other flags are reserved and should be cleared. L2CAP implementations should ignore these bits.

- *Configuration Options*

  The list of the parameters and their values to be negotiated. These are defined in Section 6 on page 288. Configuration Requests may contain no options (referred to as an empty or null configuration request) and can be used to request a response. For an empty configuration request the length field is set to 0x0004.

5.5 CONFIGURE RESPONSE (CODE 0X05)

Configure Response packets MUST be sent in reply to Configuration Request packets. Each configuration parameter value (if any is present) in a Configuration Response reflects an “adjustment” to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding Configuration Request. Thus, for example, if a configuration parameter in a Configuration Request relates to traffic flowing from device A to device B, the sender of the Configuration Response will only adjust (if needed) this value again for the same traffic flowing from device A to device B. The options sent in the Response depend on the value in the Result field. Figure 5.8 defines the format of the Configuration Response packet.

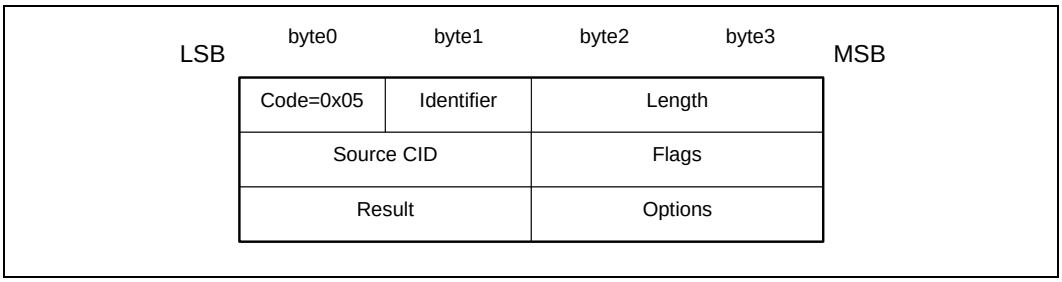

Figure 5.8: Configuration Response Packet

- *Length = 0x0006 or more octets*
- *Source CID (SCID) : 2 octets*

  The field contains the channel end-point on the device receiving this Response packet.. The device receiving the Response must check that the Identifier field matches the same field in the corresponding configuration request command and the SCID matches its local CID paired with the original DCID.
- *Flags: 2 octets*

  Figure 5.9 displays the two-octet Flags field. Note the most significant bit is shown on the left.

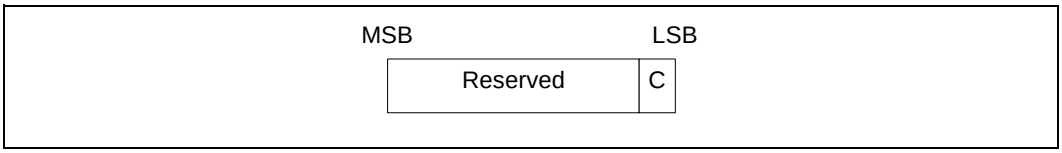

Figure 5.9: Configuration Response Flags field format

C – more configuration responses will follow when set to 1. This flag indicates that the parameters included in the response are a partial subset of parameters being sent by the device sending the Response packet.

Other flags are reserved and should be cleared. L2CAP implementations should ignore these bits.

- *Result: 2 octets*

  The Result field indicates whether or not the Request was acceptable. See Table 5.7 for possible result codes.

Result	Description
0x0000	Success
0x0001	Failure – unacceptable parameters
0x0002	Failure – rejected (no reason provided)
0x0003	Failure – unknown options
Other	RESERVED

Table 5.7: Configuration Response Result codes

- *Configuration Options*

  This field contains the list of parameters being negotiated. These are defined in Section 6 on page 288. On a successful result, these parameters contain the return values for any wild card parameters (e.g., “do not care”) contained in the request.

  On an unacceptable parameters failure (Result=0x0001) the rejected parameters should be sent in the response with the values that would have been accepted if sent in the original request. Any missing configuration parameters are assumed to have their most recently (mutually) accepted values and they too can be included in the Configuration Response if need to be changed. Recall that, each configuration parameter is one-directional and relative to the direction implied by the sender of a Configuration Request. Thus, if the sender of the Configuration Response needs to establish the value of a configuration parameter in the opposite direction than the one implied by an original Configuration Request, a new Configuration Request with the desired value of the configuration parameter in it needs to be sent in the direction opposite the one used for the original Connection Request.

  On an unknown option failure (Result=0x0003), the option types not understood by the recipient of the Request must be included in the Response. Note that hints (defined in Section 6 on page 288), those options in the Request that are skipped if not understood, must not be included in the Response and must not be the sole cause for rejecting the Request.

  The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is left to the implementation.

5.6 DISCONNECTION REQUEST (CODE 0x06)

Terminating an L2CAP channel requires that a disconnection request packet be sent and acknowledged by a disconnection response packet. Disconnection is requested using the signalling channel since all other L2CAP packets sent to the destination channel automatically get passed up to the next protocol layer. Figure 5.10 displays a disconnection packet request. The receiver must ensure both source and destination CIDs match before initiating a connection disconnection.

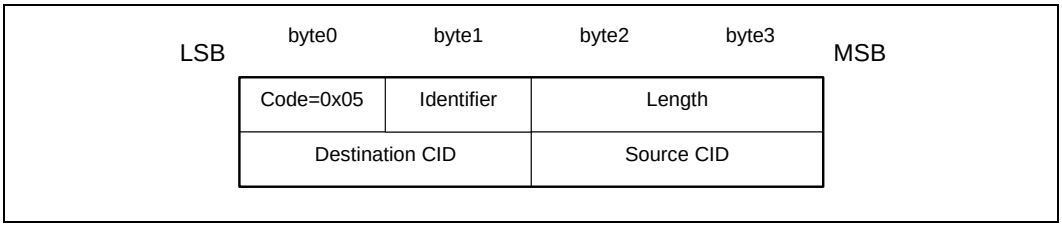

Figure 5.10: Disconnection Request Packet

- *Length = 0x0004 octets*
- *Destination CID (DCID) : 2 octets*

  This field specifies the end-point of the channel to be shutdown on the device receiving this request..
- *Local CID (LCID) : 2 octets*

  This field specifies the end-point of the channel to be shutdown on the device sending this request.

  The SCID and DCID are relative to the sender of this request and must match those of the channel to be disconnected. If the DCID is not recognized by the receiver of this message, a CommandReject message with “invalid CID” result code must be sent in response. If the receivers finds a DCID match but the SCID fails to find the same match, the request should be silently discarded.

5.7 DISCONNECTION RESPONSE (CODE 0x07)

Disconnection responses should be sent in response to each disconnection request.

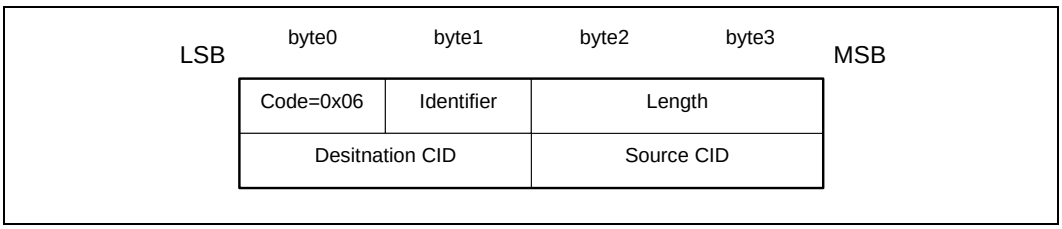

Figure 5.11: Disconnection Response Packet

- *Length = 0x0004 octets*
- *Destination CID (DCID) : 2 octets*

  This field identifies the channel end-point on the device sending the response.
- *Source CID (SCID)*

  This field identifies the channel end-point on the device receiving the response.

  The DCID and the SCID(which are relative to the sender of the request), and the Identifier fields must match those of the corresponding disconnection request command. If the CIDs do not match, the response should be silently discarded at the receiver.

5.8 ECHO REQUEST (CODE 0x08)

Echo requests are used to solicit a response from a remote L2CAP entity. These requests may be used for testing the link or passing vendor specific information using the optional data field. L2CAP entities MUST respond to well-formed Echo Request packets with an Echo Response packet. The Data field is optional and implementation dependent. L2CAP entities should ignore the contents of this field.

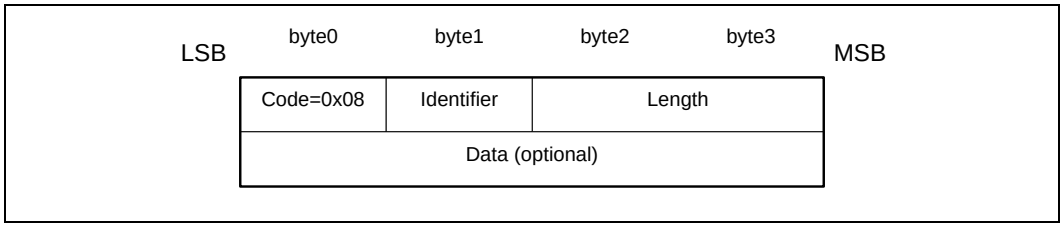

Figure 5.12: Echo Request Packet

5.9 ECHO RESPONSE (CODE 0x09)

Echo responses are sent upon receiving Echo Request packets. The identifier in the response MUST match the identifier sent in the Request. The optional and implementation dependent data field may contain the contents of the data field in the Request, different data, or no data at all.

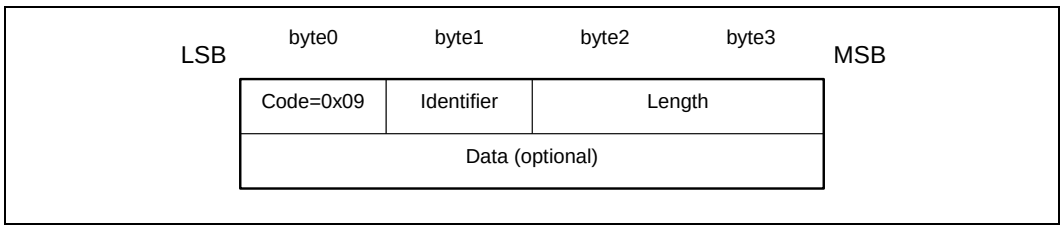

Figure 5.13: Echo Response Packet

5.10 INFORMATION REQUEST

Information requests are used to solicit implementation-specific information from a remote L2CAP entity. L2CAP entities MUST respond to well-formed Information Request packets with an Information Response packet.

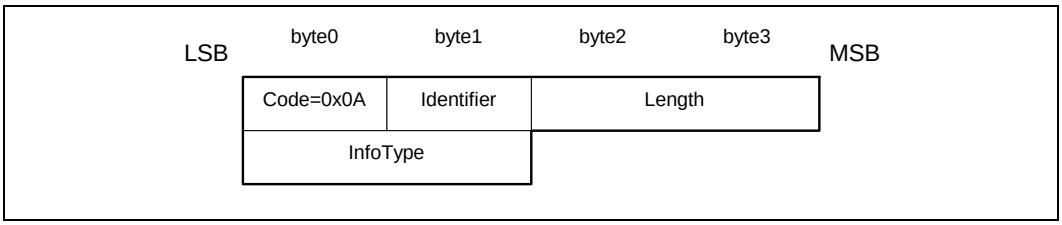

Figure 5.14: Information Request Packet

- *Length = 0x0002 octets*
- *InfoType: 2 octets*

  The InfoType defines the type of implementation-specific information being solicited.

Value	Description
0x0001	Connectionless MTU
Other	Reserved

Table 5.8: InfoType definitions

5.11 INFORMATION RESPONSE

Information responses are sent upon receiving Information Request packets. The identifier in the response MUST match the identifier sent in the Request. The optional data field may contain the contents of the data field in the Request, different data, or no data at all.

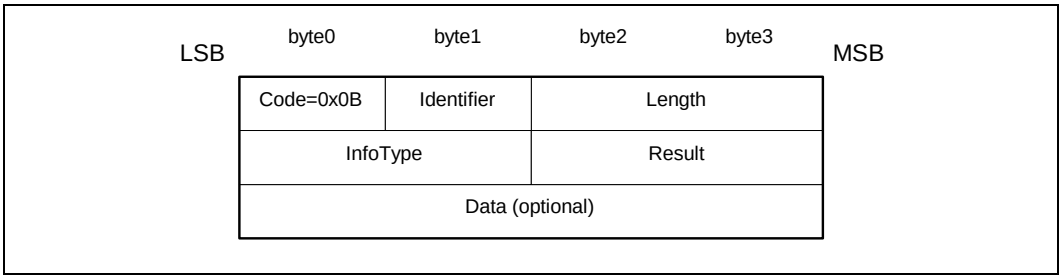

Figure 5.15: Information Response Packet

- *InfoType: 2 octets*

  Same value sent in the request.

- *Result: 2 octets*

  The Result contains information about the success of the request. If successful, the data fields contain the information as specified in Table 5.10. If unsuccessful, no data should be returned.

Value	Description
0x0000	Success
0x0001	Not supported
Other	Reserved

Table 5.9: Information Response Result values

- *Data: 0 or more octets*

  The contents of the Data field depends on the InfoType. For the Connection-MTU request, the data field contains the remote entity's 2-octet acceptable connectionless MTU.

InfoType	Data	Data Length (in octets)
0x0001	Connectionless MTU	2

Table 5.10: Information Response Data fields

6 CONFIGURATION PARAMETER OPTIONS

Options are a mechanism to extend the ability to negotiate different connection requirements. Options are transmitted in the form of information elements comprised an option type, an option length, and one or more option data fields. Figure 6.1 illustrates the format of an option.

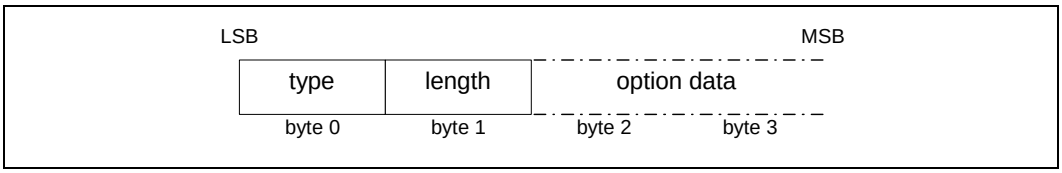

Figure 6.1: Configuration option format

- *Option type: 1 octet*

  The option type field defines the parameter being configured. The most significant bit of the type determines the action taken if the option is not recognized. Option types whose MSB is set to 1 are referred to as “hints”.

  0 – refuse the configuration request

  1 – skip the option and continue processing

- *Length: 1 octet*

  The length field defines the number of octets in the option payload. So an option type with no payload has a length of 0.

- *Option data*

  The contents of this field are dependent on the option type.

6.1 MAXIMUM TRANSMISSION UNIT (MTU)

This option specifies the payload size the sender is capable of accepting. The type is 0x01, and the payload length is 2 bytes, carrying the two-octet MTU size value as the only information element (see Figure 6.2 on page 289).

Since all L2CAP implementations are capable to support a minimum L2CAP packet size, see Section 4 on page 272, MTU is not really a negotiated value but rather an informational parameter to the remote device that the local device can accommodate in this channel an MTU larger than the minimum required. In the unlikely case that the remote device is only willing to send L2CAP packets in this channel that are larger than the MTU announced by the local device, then this Configuration Request will receive a negative response in which the remote device will include the value of MTU that is indented to transmit. In this case, it is implementation specific on whether the local device will continue the configuration process or even maintain this channel.

The remote device in its positive Configuration Response will include the actual MTU to be used on this channel for traffic flowing into the local device which is

minimum{ MTU in configReq, outgoing MTU capability of remote device }. The MTU to be used on this channel but for the traffic flowing in the opposite direction will be established when the remote device (with respect to this discussion) sends its own Configuration Request as explained in Section 5.4 on page 280.

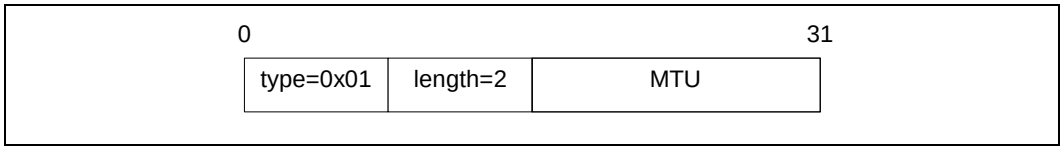

Figure 6.2: MTU Option Format

- Maximum Transmission Unit (MTU) Size: 2 octets

  The MTU field represents the largest L2CAP packet payload, in bytes, that the originator of the Request can accept for that channel. The MTU is asymmetric and the sender of the Request shall specify the MTU it can receive on this channel if it differs from the default value. L2CAP implementations must support a minimum MTU size of 48 bytes. The default value is 672 bytes[1].

6.2 FLUSH TIMEOUT OPTION

This option is used to inform the recipient of the amount of time the originator's link controller / link manager will attempt to successfully transmit an L2CAP segment before giving up and flushing the packet. The type is 0x02 and the payload size is 2 octets.

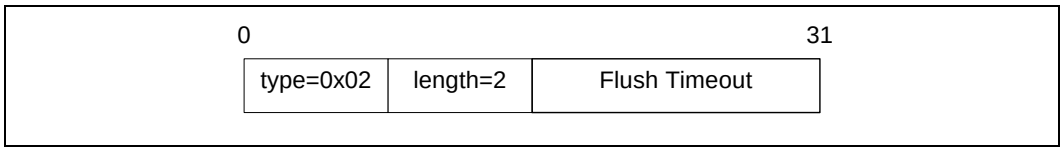

Figure 6.3: Flush Timeout

- *Flush Timeout*

  This value represents units of time measured in milliseconds. The value of 1 implies no retransmissions at the Baseband level should be performed since the minimum polling interval is 1.25 ms. The value of all 1's indicates an infinite amount of retransmissions. This is also referred to as "reliable channel". In this case, the link manager shall continue retransmitting a segment until physical link loss occurs. This is an asymmetric value and the sender of the Request shall specify its flush timeout value if it differs from the default value of 0xFFFF.

1. The default MTU was selected based on the payload carried by two Baseband DH5 packets (2*341=682) minus the Baseband ACL headers (2*2=4) and L2CAP header (6).

6.3 QUALITY OF SERVICE (QOS) OPTION

This option specifies a flow specification (flowSpec) similar to RFC 1363 [17]. If no QoS configuration parameter is negotiated the link should assume the default parameters discussed below. The QoS option is type 0x04.

When included in a Configuration Request, this option describes the outgoing traffic flow from the device sending the request to the device receiving it. When included in a positive Configuration Response, this option describes the incoming traffic flow agreement as seen from the device sending the response. When included in a negative Configuration Response, this option describes the preferred incoming traffic flow from the perspective of the device sending the response.

L2CAP implementations are only required to support ”Best Effort” service, support for any other service type is optional. Best Effort does not require any guarantees. If no QoS option is placed in the request, Best Effort must be assumed. If any QoS guarantees are required then a QoS configuration request must be sent.

The remote device places 1information that depends on the value of the result field, see Section 5.5 on page 282, in its Configuration Response. If the result is “Success,” the response may include any specific values of the outgoing flowspec of possible wild card (e.g., “do not care”) parameters contained (or implied) in the request. If the result is “Failure – unacceptable parameters,” the response may include a list of outgoing flowspec parameters and parameter values that would make a new Connection Request from the local device acceptable by the remote device. Both explicitly referenced in a Configuration Request or implied configuration parameters can be included in a Configuration Response. Recall that any missing configuration parameters from a Configuration Request are assumed to have their most recently (mutually) accepted values.

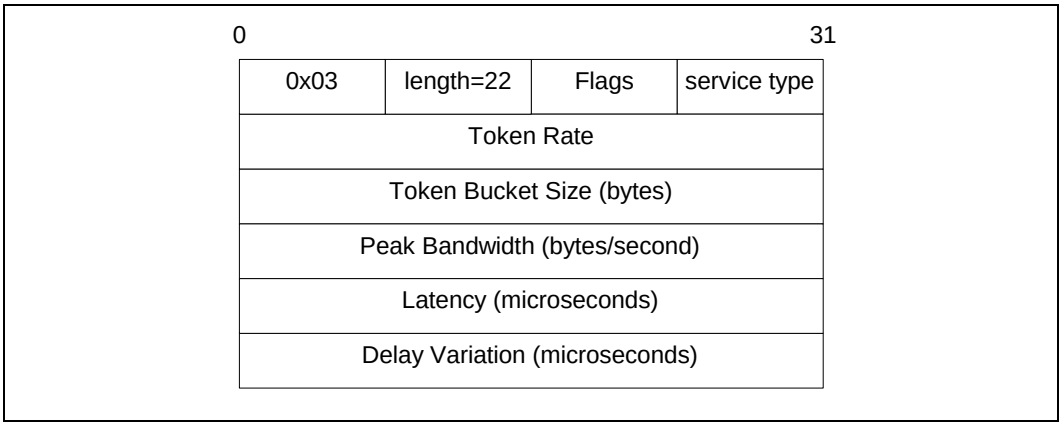

Figure 6.4: Quality of Service Flow Specification

- *Flags: 1 octet*

  Reserved for future use and must be set to 0.

- *Service type: 1 octet*

  This field indicates the level of service required. Table 6.1 defines the different services available. If “No traffic” is selected, the remainder of the fields may be ignored because there is no data being sent across the channel in the outgoing direction.

  If “Best effort”, the default value, is selected, the remaining fields should be treated as hints by the remote device. The remote device may choose to ignore the fields, try to satisfy the hint but provide no response (QoS option omitted in the Response message), or respond with the settings it will try to meet.

Value	Description
0x00	No traffic
0x01	Best effort (Default)
0x02	Guaranteed
Other	Reserved

Table 6.1: Service type definitions

- *Token Rate: 4 octets*

  The value of this field represents the rate at which traffic credits are granted in bytes per second. An application may send data at this rate continuously. Burst data may be sent up to the token bucket size (see below). Until that data burst has been drained, an application must limit itself to the token rate. The value 0x00000000 indicates no token rate is specified. This is the default value and implies indifference to token rate. The value 0xFFFFFFFF represents a wild card matching the maximum token rate available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value is a hint that the application wants as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

- *Token Bucket Size: 4 octets*

  The value of this field represents the size of the token bucket in bytes. If the bucket is full, then applications must either wait or discard data. The value of 0x00000000 represents no token bucket is needed; this is the default value. The value 0xFFFFFFFF represents a wild card matching the maximum token bucket available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value indicates the application wants a bucket as big as possible. For Guaranteed service the value represents the maximum buffer space available at the time of the request.

- *Peak Bandwidth: 4 octets*

  The value of this field, expressed in bytes per second, limits how fast packets may be sent back to back from applications. Some intermediate systems can take advantage of this information resulting in more efficient resource

allocation. The value of 0x00000000 states that the maximum bandwidth is unknown, which is the default value.

- *Latency: 4 octets*

  The value of this field represents the maximum acceptable delay between transmission of a bit by the sender and its initial transmission over the air, expressed in microseconds. The precise interpretation of this number depends on the level of guarantee specified in the Class of Service. The value 0xFFFFFFFF represents a do not care and is the default value.

- *Delay Variation: 4 octets*

  The value of this field is the difference, in microseconds, between the maximum and minimum possible delay that a packet will experience. This value is used by applications to determine the amount of buffer space needed at the receiving side in order to restore the original data transmission pattern. The value 0xFFFFFFFF represents a do not care and is the default value.

6.4 CONFIGURATION PROCESS

Negotiating the channel parameters involves three steps:

1. Informing the remote side of the non-default parameters that the local side will accept
2. Having the remote side agreeing or disagreeing to these values (including the default ones); steps (1) and (2) may iterate as needed
3. Repeat steps (1) and (2) for the reverse direction from the (previous) remote side to the (previous) local side.

This process can be abstracted into a Request negotiation path and a Response negotiation path.

6.4.1 Request Path

The Request Path negotiates the incoming MTU, flush timeout, and outgoing flowspec. Table 6.2 defines the configuration options that may be placed in the Configuration Request message and their semantics.

Parameter	Description
MTU.	Incoming MTU information
FlushTO	Outgoing flush timeout
OutFlow	Outgoing flow information.

Table 6.2: Parameters allowed in Request

6.4.2 Response Path

The Response Path negotiates the outgoing MTU (remote side's incoming MTU), the remote side's flush timeout, and incoming flowspec (remote side's outgoing flowspec). If a request-oriented parameter is not present in the Request message (reverts to default value), the remote side may negotiate for a non-default value by including the proposed value in a negative Response message.

Parameter	Description
MTU	Outgoing MTU information.
FlushTO	Incoming flush timeout.
InFlow	Incoming flow information.

Table 6.3: Parameters allowed in Response

6.4.3 Configuration State Machine

The configuration state machine shown below depicts two paths. Before leaving the CONFIG state and moving into the OPEN state, both paths must reach closure. The request path requires the local device to receive a positive response to reach closure while the response path requires the local device to send a positive response to reach closure.

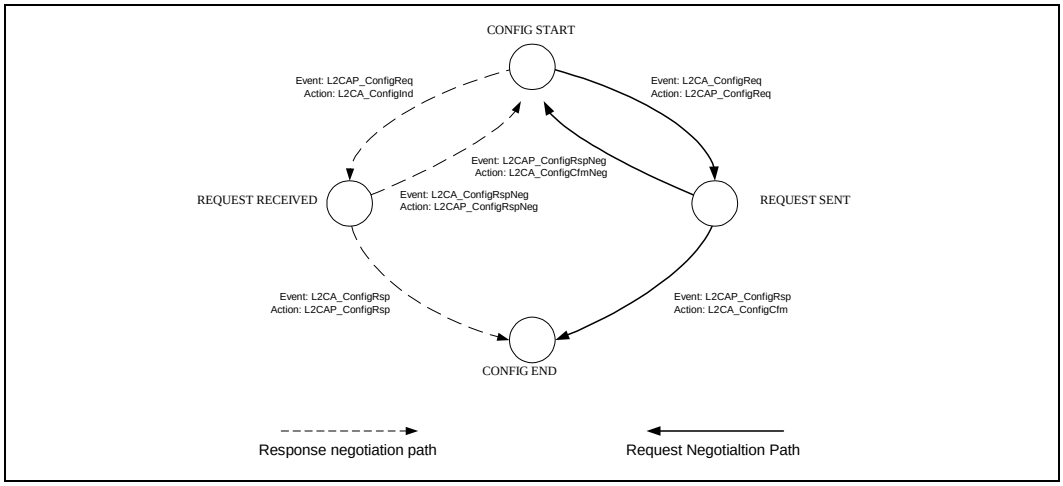

Figure 6.5: Configuration State Machine

"Appendix A: Configuration MSCs" on page 316 provides some configuration examples.

7 SERVICE PRIMITIVES

This section presents an abstract description of the services offered by L2CAP in terms of service primitives and parameters. The service interface is required for testing. The interface is described independently of any platform specific implementation. All data values use little endian byte ordering.

7.1 EVENT INDICATION

Service	Input Parameters	Output Parameters
EventIndication	Event, Callback	Result

Description:

The use of this primitive requests a callback when the selected indication Event occurs.

Input Parameters:

Event *Type: uint* *Size: 2 octets*

Value	Description
0x00	Reserved
0x01	L2CA_ConnectInd
0x02	L2CA_ConfigInd
0x03	L2CA_DisconnectInd
0x04	L2CA_QoSViolationInd
other	Reserved for future use

Callback *Type: function* *Size: N/A*

Event	Callback Function Input Parameters
L2CA_ConnectInd	BD_ADDR, CID, PSM, Identifier
L2CA_ConfigInd	CID, OutMTU, InFlow, FlushTO
L2CA_DisconnectInd	CID
L2CA_QoSViolationInd	BD_ADDR

7.1.1 L2CA_ConnectInd Callback

This callback function includes the parameters for the address of the remote device that issued the connection request, the local CID representing the channel being requested, the Identifier contained in the request, and the PSM value the request is targeting.

7.1.2 L2CA_ConfigInd Callback

This callback function includes the parameters indicating the local CID of the channel the request has been sent to, the outgoing MTU size (maximum packet that can be sent across the channel) and the flowspec describing the characteristics of the incoming data. All other channel parameters are set to their default values if not provided by the remote device.

7.1.3 L2CA_DisconnectInd Callback

This callback function includes the parameter indicating the local CID the request has been sent to.

7.1.4 L2CA_QoSViolationInd Callback

This callback function includes the parameter indicating the address of the remote Bluetooth device where the QoS contract has been violated.

7.2 CONNECT

Service	Input Parameters	Output Parameters
L2CA_ConnectReq	PSM, BD_ADDR	LCID, Result, Status

Description:

This primitive initiates the sending of an L2CA_ConnectReq message and blocks until a corresponding L2CA_ConnectCfm(Neg) or L2CA_TimeOutInd message is received.

The use of this primitive requests the creation of a channel representing a logical connection to a physical address. Input parameters are the target protocol (*PSM*) and remote device's 48-bit address (*BD_ADDR*). Output parameters are the local CID *(LCID)* allocated by the local L2CAP entity, and *Result* of the request. If the *Result* indicates success, the *LCID* value contains the identification of the local endpoint. Otherwise the *LCID* returned should be set to 0. If *Result* indicates a pending notification, the *Status* value may contain more information of what processing is delaying the establishment of the connection. Otherwise the *Status* value should be ignored.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Target PSM provided for the connection.

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device.

Output Parameters:

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel if Result = 0x0000, otherwise set to 0.

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Connection successful and the CID identifies the local endpoint. Ignore Status parameter
0x0001	Connection pending. Check Status parameter for more information.
0x0002	Connection refused because no service for the PSM has been registered.
0x0003	Connection refused because the security architecture on the remote side has denied the request.
0xEEEE	Connection time out occurred. This is a result of a timer expiration indication being included in the connection confirm message

Status *Type: uint* *Size: 2 octets*

Value	Description
0x0000	No further information.
0x0001	Authentication pending.
0x0002	Authorisation pending.

7.3 CONNECT RESPONSE

Service	Input Parameters	Output Parameters
L2CA_ConnectRsp	BD_ADDR, Identifier, LCID, Response, Status	Result

Description:

This primitive represents the L2CA_ConnectRsp.

The use of this primitive issues a response to a connection request event indication. Input parameters are the remote device's 48-bit address, Identifier sent in the request, local CID, the Response code, and the Status attached to the Response code. The output parameters include the local CID and the Result of the service request.

This primitive must be called no more than once after receiving the callback indication. This primitive returns once the local L2CAP entity has validated the

request. A successful return does indicate the response has been sent over the air interface.

Input Parameters:

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

Identifier *Type: uint* *Size: 1 octets*

Value	Description
0xXX.	This value must match the value received in the L2CA_ConnectInd event described in Section 7.1.1 on page 294

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

Response *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Connection successful
0x0001	Connection pending
0x0002	Connection refused – PSM not supported
0x0003	Connection refused – security block
0x0004	Connection refused – no resources available
0xXXXX	Other connection response code

Status *Type: uint* *Size: 2 octets*

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorisation pending
0xXXXX	Other status code

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response successfully sent
0x0001	Failure to match any outstanding connection request

7.4 CONFIGURE

Service	Input Parameters	Output Parameters
L2CA_ConfigReq	CID, InMTU, OutFlow, FlushTO, LinkTO	Result, InMTU, OutFlow, FlushTO

Description:

This primitive initiates the sending of an L2CA_ConfigReq message and blocks until a corresponding L2CA_ConfigCfm(Neg) or L2CA_TimeOutInd message is received.

The use of this primitive requests the initial configuration (or reconfiguration) of a channel to a new set of channel parameters. Input parameters are the local CID endpoint, new incoming receivable MTU (InMTU), new outgoing flow specification, and flush and link timeouts. Output parameters composing the L2CA_ConfigCfm(Neg) message are the Result, accepted incoming MTU(InMTU), the remote side's flow requests, and flush and link timeouts. Note that the output results are returned only after the local L2CAP entity transitions out of the CONFIG state (even if this transition is back to the CONFIG state).

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	(Local CID.

InMTU *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit this channel can accept.

OutFlow *Type: Flow* *Size: x octets*

Value	Description
flowspec	Quality of service parameters dealing with the traffic characteristics of the outgoing data flow

FlushTO

Value	Description
0xXXXX	Number of milliseconds to wait before an L2CAP packet that cannot be acknowledged at the physical layer is dropped.
0x0000	Request to use the existing flush timeout value if one exists, otherwise the default value (0xFFFF) will be used.
0x0001	Perform no retransmissions at the Baseband layer.
0xFFFF	Perform retransmission at the Baseband layer until the link timeout terminates the channel.

LinkTO

Value	Description
0xXXXX	Number of milliseconds to wait before terminating an unresponsive link.

Output Parameters:

Result

Value	Description
0x0000	Configuration is successful. Parameters contain agreed upon values.
0x0001	Failure – invalid CID
0x0002	Failure – unacceptable parameters
0x0003	Failure – signalling MTU exceeded
0x0004	Failure – unknown options
0xEEEE	Configuration time out occurred. This is a result of a timer expiration indication being included in the configuration confirm.

InMTU

Value	Description
0xXXXX	Maximum transmission unit that should be sent by this unit across this channel.

OutFlow

Value	Description
FlowSpec	Quality of service parameters dealing with the traffic characteristics of the agreed upon outgoing data flow if Result is successful. Otherwise this represents the requested Quality of Service.

FlushTO

Value	Description
0xXXXX	Number of milliseconds before an L2CAP packet that cannot be acknowledged at the physical layer is dropped. This value is informative of the remote devices FlushTO parameter.

7.5 CONFIGURATION RESPONSE

Service	Input Parameters	Output Parameters
L2CA_ConfigRsp	CID, OutMTU, InFlow,	Result

Description:

This primitive represents the L2CA_ConfigRsp.

The use of this primitive issues a response to a configuration request event indication. Input parameters include the local CID of the endpoint being configured, outgoing transmit MTU (OutMTU), the outgoing flowspec, the incoming flowspec, and the local endpoint's flush timeout. The output parameter is the Result value.

Input Parameters:

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Local channel identifier.

OutMTU *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit this channel will send.

InFlow *Type: Flow* *Size: x octets*

Value	Description
FlowSpec	Quality of service parameters dealing with the traffic characteristics of the incoming data flow

Output Parameters:

Result

Value	Description
0x0000	Configuration is successful. Parameters contain agreed upon values.
0x0001	Configuration failed – unacceptable parameters
0x0002	Configuration failed - rejected
0x0003	Configuration failed – invalid CID
0x0004	Configuration failed – unknown options
0xXXXX	Reserved

7.6 DISCONNECT

Service	Input Parameters	Output Parameters
L2CA_DisconnectReq	CID	Result

Description:

This primitive represents the L2CAP_DisconnectionReq and the returned output parameters represent the corresponding L2CA_DisconnectionCfmor the RTX timer expiration.

The use of this primitive requests the disconnection of the channel. Input parameter is the *CID* representing the local channel endpoint. Output parameter composing the L2CA_DisconnectCfm is *Result*. *Result* is zero if a L2CA_DiconnectCfm is received, otherwise a non-zero value is returned.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Disconnection successful. This is a result of the receipt of a disconnection response message.
0xEEEE	Disconnection timeout occurred. This is a result of a timer expiration indication being included in the disconnection confirm message.

7.7 WRITE

Service	Input Parameters	Output Parameters
L2CA_WriteData	CID, Length, OutBuffer	Size, Result

Description:

The use of this primitive requests the transfer of data across the channel. If the length of the data exceeds the OutMTU then only the first OutMTU bytes are sent This command may be used for both connection-oriented and connection-less traffic.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the buffer where data to be transmitted are stored.

OutBuffer *Type: pointer* *Size: N/A*

Value	Description
N/A	Address of the input buffer used to store the message.

Output Parameters:

Size *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	The number of bytes transferred.

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful write.
0x0001	Error – Flush timeout expired
0x0002	Error – Link termination (perhaps this should be left to the indication)

7.8 READ

Service	Input Parameters	Output Parameters
L2CA_ReadData	CID, Length, InBuffer	Result

Description:

The use of this primitive requests for the reception of data. This request returns when data is available or the link is terminated. The data returned represents a single L2CAP payload. If not enough data is available the command will block until the data arrives or the link is terminated. If the payload is bigger than the buffer, only the portion of the payload that fits into the buffer will be returned and the remainder of the payload will be discarded. This command may be used for both connection-oriented and connectionless traffic.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	CID.

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the buffer where received data are to be stored.

InBuffer *Type: pointer* *Size: N/A*

Value	Description
N/A	Address of the buffer used to store the message.

Output parameters:

Result

Value	Description
0x0000	Success

7.9 GROUP CREATE

Service	Input Parameters	Output Parameters
L2CA_GroupCreate	PSM	CID

Description:

The use of this primitive requests the creation of a CID to represent a logical connection to multiple devices. Input parameter is the *PSM* value that the outgoing connectionless traffic is labelled with and the filter used for incoming traffic. Output parameter is the *CID* representing the local endpoint. On creation, the group is empty but incoming traffic destined for the PSM value is readable.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Protocol/service multiplexer value.

Output Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

7.10 GROUP CLOSE

Service	Input Parameters	Output Parameters
L2CA_GroupClose	CID	Result

Description:

The use of this primitive closes down a Group.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful closure off the channel.
0x0001	Invalid CID

7.11 GROUP ADD MEMBER

Service	Input Parameters	Output Parameters
L2CA_GroupAddMember	CID, BD_ADDR	Result

Description:

The use of this primitive requests the addition of a member to a group. The input parameter includes the CID representing the group and the BD_ADDR of the group member to be added. The output parameter Result confirms the success or failure of the request.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXX	Remote device address.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success.
0x0001	Failure to establish connection to remote device.
Other	Reserved.

7.12 GROUP REMOVE MEMBER

Service	Input Parameters	Output Parameters
L2CA_GroupRemoveMember	CID, BD_ADDR	Result

Description:

The use of this primitive requests the removal of a member from a group. The input parameters include the CID representing the group and BD_ADDR of the group member to be removed. The output parameter Result confirms the success or failure of the request.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address device to be removed.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success.
0x0001	Failure – device not a member of the group.
Other	Reserved.

7.13 GET GROUP MEMBERSHIP

Service	Input Parameters	Output Parameters
L2CA_GroupMembership	CID	Result, N, BD_ADDR_Lst

Description:

The use of this primitive requests a report of the members of a group. The input parameter CID represents the group being queried. The output parameter Result confirms the success or failure of the operation. If the Result is successful, BD_ADDR_Lst is a list of the Bluetooth addresses of the N members of the group.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success.
0x0001	Failure – group does not exist.
Other	Reserved.

N *Type: uint* *Size: 2 octets*

Value	Description
0x0000-0xFFFF	The number of devices in the group identified by the channel end-point CID. If Result indicates failure, N should be set to 0.

BD_ADDR_List *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	List of N unique Bluetooth addresses of the devices in the group identified by the channel end-point CID. If Result indicates failure, the all-zero address is the only address that should be returned.

7.14 PING

Service	Input Parameters	Output Parameters
L2CA_Ping	BD_ADDR, ECHO_DATA	Result, ECHO_DATA

Description:

This primitive represents the L2CA_EchoReq and the output parameters represent the L2CA_EchoRep.

Input Parameters:

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response received.
0x0001	Timeout occurred.

7.15 GETINFO

Service	Input Parameters	Output Parameters
L2CA_GetInfo	BD_ADDR, InfoType	Result, InfoData

Description:

This primitive represents the L2CA_InfoReq and the output parameters represent the L2CA_InfoRep.

Input Parameters:

BD_ADDR *Type: unit* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device.

InfoType *Type: unit* *Size: 2 octets*

Value	Description
0x0001	Maximum connectionless MTU size

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response received.
0x0001	Not supported.
0x0002	Informational PDU rejected, not supported by remote device.
0x0003	Timeout occurred.

InfoData *Type: uint* *Size: Variable*

For InfoType=0x0001, the output value is a two-octet field.

7.16 DISABLE CONNECTIONLESS TRAFFIC

Service	Input Parameters	Output Parameters
L2CA_DisableCLT	N, List of PSMs	Result

Description:

General request to disable the reception of connectionless packets. The input parameter is the *PSM* value indicating service that should be blocked. This command may be used to incrementally disable a set of PSM values. The use of the “invalid” PSM 0x0000 blocks all connectionless traffic. The output parameter *Result* indicates the success or failure of the command. A limited device might support only general blocking rather than PSM-specific blocks and would fail to block a single non-zero PSM value.

Input Parameters:

N *Type: uint* *Size: 2 octets*

Value	Description
0x0001-0xFFFF	The number of PSMs to be blocked. When all the PSMs are to be blocked use the reserved value N=0x0001.

List of PSMs *Type: uint* *Size: max(N,1) times 2 octets*

Value	Description
0x0000	Block all connectionless traffic.
0xXXXX	Protocol/Service Multiplexer field to be blocked.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful.
0x0001	Failure – not supported.

7.17 ENABLE CONNECTIONLESS TRAFFIC

Service	Input Parameters	Output Parameters
L2CA_EnableCLT	N, List of PSMs	Result

Description:

General request to enable the reception of connectionless packets. The input parameter is the *PSM* value indicating service that should be unblocked. This command may be used to incrementally enable a set of PSM values. The use of the “invalid” CID 0x0000 enables all connectionless traffic. The output parameter *Result* indicates the success or failure of the command. A limited device might support only general enabling rather than PSM-specific filters and would fail to enable a single non-zero PSM value.

Input Parameters:

N *Type: uint* *Size: 2 octets*

Value	Description
0x0001-0xFFFF	The number of PSMs to be unblocked blocked. When all the PSMs are to be unblocked use the reserved value N=0x0000.

PSM *Type: uint* *Size: max(N,1) times 2 octets*

Value	Description
0x0000	Enables all connectionless traffic.
0xXXXX	Protocol/Service Multiplexer field to enable.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful.
0x0001	Failure – not supported.

8 SUMMARY

The Logical Link Control and Adaptation Protocol (L2CAP) is one of two link level protocols running over the Baseband. L2CAP is responsible for higher level protocol multiplexing, MTU abstraction, group management, and conveying quality of service information to the link level.

Protocol multiplexing is supported by defining channels. Each channel is bound to a single protocol in a many-to-one fashion. Multiple channels can be bound to the same protocol, but a channel cannot be bound to multiple protocols. Each L2CAP packet received on a channel is directed to the appropriate higher level protocol.

L2CAP abstracts the variable-sized packets used by the Baseband Protocol (page 33). It supports large packet sizes up to 64 kilobytes using a low-overhead segmentation and reassembly mechanism.

Group management provides the abstraction of a group of units allowing more efficient mapping between groups and members of the Bluetooth piconet. Group communication is connectionless and unreliable. When composed of only a pair of units, groups provide connectionless channel alternative to L2CAP's connection-oriented channel.

L2CAP conveys QoS information across channels and provides some admission control to prevent additional channels from violating existing QoS contracts.

9 REFERENCES

[1] Bluetooth Special Interest Group, “Part B: Baseband Specification”, Specification of the Bluetooth System, Version 0.9, April 12, 1999.

[2] Bluetooth Special Interest Group, “Part C: Link Manager”, Specification of the Bluetooth System, Version 0.9, April 12, 1999.

[3] Bluetooth Special Interest Group, “Part F1: RFCOMM with TS 07.10”, Specification of the Bluetooth System, Version 0.9, April 12, 1999.

[4] Bluetooth Special Interest Group, “Section H:1 Bluetooth Host Controller Functional Interface Specification”, Specification of the Bluetooth System, Version 0.9, April 12, 1999.

[5] Bluetooth Special Interest Group, “Service Discovery Protocol”, Specification of the Bluetooth System, Version 0.9, April 12, 1999.

[6] Bluetooth Special Interest Group, “Assigned Numbers”, Bluetooth Specification Document, Version 0.9, April 12, 1999.

[7] Bluetooth Special Interest Group, “Telephony Control Protocol”, Specification of the Bluetooth System, Version 0.9, April 12, 1999.

[8] Internet Engineering Task Force, “Point-to-Point Protocol”, RFC 1661.

[9] AAL5, ITU 1663, 1994

[10] Real-time Transport Protocol, IETF, RFC1889

[11] RSVP, Internet-draft, IETF, 1997

[12] GSM 07.10, draft, ETSI, 1997

[13] IrDA document set, IrDA, 1997.

[14] Universal Serial Bus Specification, Revision 1.0, 1996.

[15] Plug and Play System Architecture, MindShare Inc., 1995.

[16] Microsoft documents on PnP, Microsoft Corporation.

[17] Internet Engineering Task Force, “A Proposed Flow Specification”, RFC 1363, September 1992.

10 LIST OF FIGURES

11 LIST OF TABLES

TERMS AND ABBREVIATIONS

Baseband	Baseband Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IrDA	Infra-red Data Association
L_CH	Logical Channel
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
MTU	Maximum Transmission Unit.
PPP	Point-to-Point Protocol
Reliable	Characteristic of an L2CAP channel that has an infinite flush timeout.
RFC	Request For Comments
SAR	Segmentation and Reassembly

APPENDIX A: CONFIGURATION MSCS

The examples in this appendix describe a sample of the multiple possible configuration scenarios that might occur. Currently, these are provided as suggestions and may change in the next update of the Specification.

Figure I illustrates the basic configuration process. In this example, the devices exchange MTU information. All other values are assumed to be default.

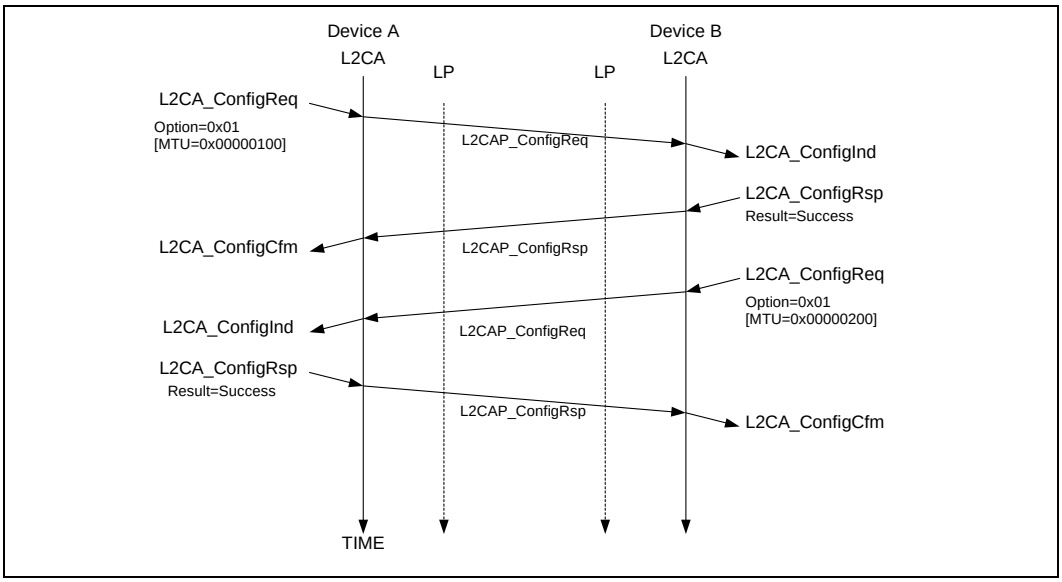

Figure I: Basic MTU exchange

Figure II on page 317 illustrates how two devices interoperate even though one device supports more options than the other does. Device A is an upgraded version. It uses a hypothetically defined option type 0x20 for link level security. Device B rejects the command using the Configuration Response packet with result “unknown parameter” informing Device A that option 0x20 is not understood. Device A then resends the request omitting option 0x20. Device B notices that it does not need to such a large MTU and accepts the request but includes in the response the MTU option informing Device A that Device B will not send an L2CAP packet with a payload larger than 0x80 octets over this channel. On receipt of the response, Device A could reduce the buffer allocated to hold incoming traffic.

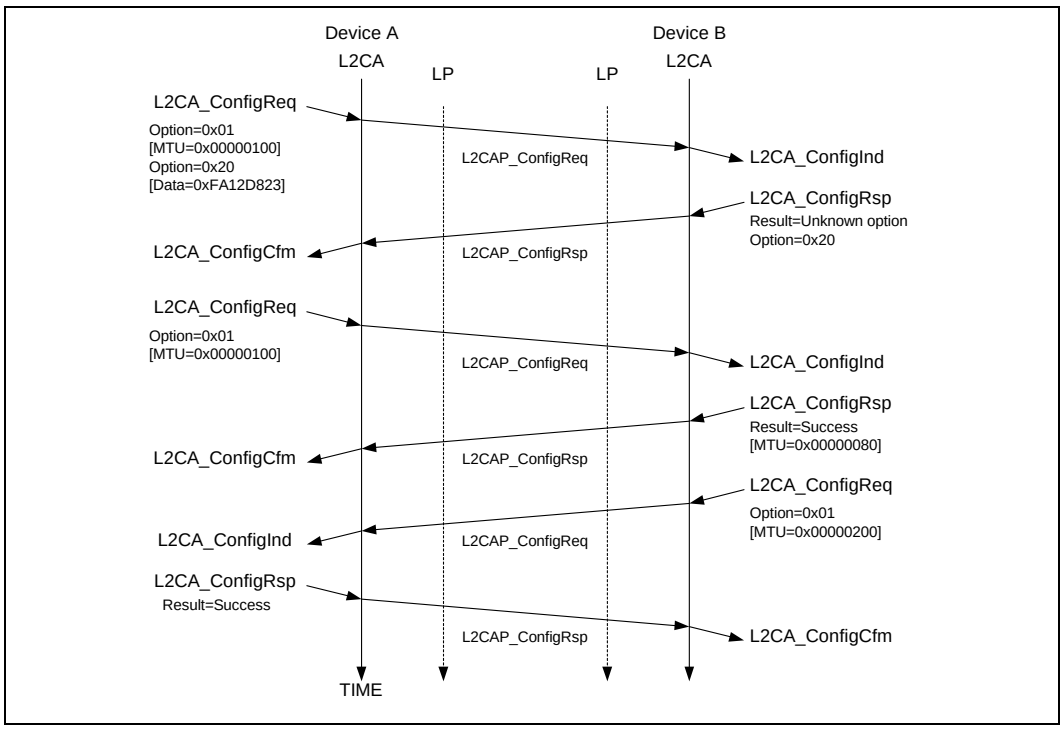

Figure II: Dealing with Unknown Options

Figure III on page 318 illustrates an unsuccessful configuration request. There are two problems described by this example. The first problem is the configuration request is placed in an L2CAP packet that cannot be accepted by the remote device due to its size. The remote device informs the sender of this problem using the Command Reject message.

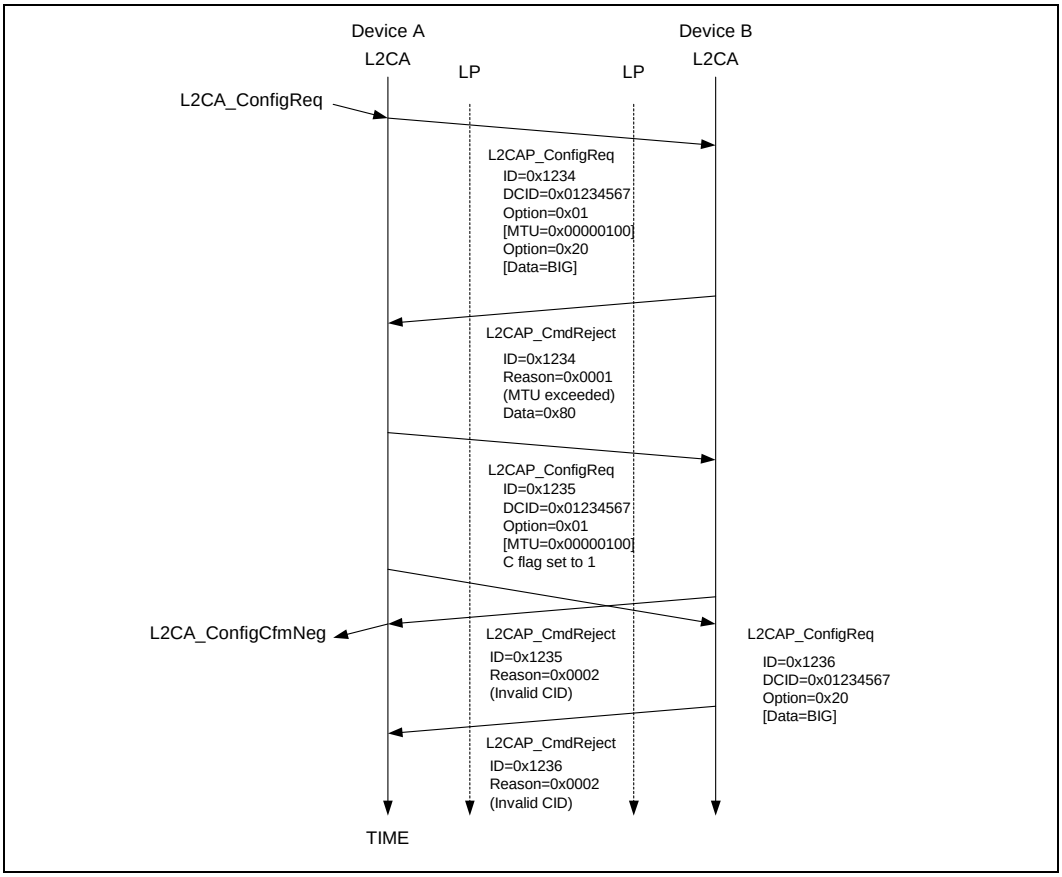

Figure III: Unsuccessful Configuration Request

APPENDIX B: IMPLEMENTATION GUIDELINES

This section contains some guidelines for implementations. These guidelines are not part of the compliance tests. At the moment they are simply suggestions on how to solve some difficult problems.

RTX TIMER

Implementations should not start this timer on an L2CAP Connection Request packet unless the physical link has been established. Otherwise the Baseband paging mechanism might increase the cost of the request beyond that of the minimal timeout value. If an implementation performs some form of security checks it is recommended the connection pending response be sent back prior to any consultation with a security manager that might perform Baseband authentication commands. If any security check requires user interaction, the link might time out waiting for the user to enter a PIN.

QOS MAPPING TO LM AND L2CAP IMPLEMENTATIONS

Token Rate

The Link Manager (LM) should ensure data is removed from the transmission buffer at this rate. The LM should ensure the polling interval is fast enough to support this data rate. The polling interval should be adjusted if the packet type changes. If the buffer overflows, and the service type is Guaranteed, a QoS violation should be reported. If the service type is Best Effort and a Token Rate was non-zero a QoS violation should also be reported.

Given a Token Rate of 0xFFFFFFFF and Service Type of Guaranteed, the LM should refuse any additional connections from remote devices and disable all periodic scans.

Token Bucket Size

L2CAP implementations should ensure a buffer meeting the size request is allocated for the channel. If no buffer is available and the service type is Guaranteed the request should be rejected. If no appropriately sized buffer is available and the service type is Best Effort, the largest available buffer should be allocated.

Peak Bandwidth

If the token bucket buffer overflows, a QoS violation should be raised.

Latency

The LM should ensure the polling interval is at least this value. If the polling interval necessary to support the token rate is less than this value, the smaller interval should be used. If this interval cannot be supported, a QoS violation should be raised.

Delay Variation

The LM may ignore this value because there is no clear mapping between L2CAP packet delays and the necessary polling interval without requiring the LM to comprehend the length field in L2CAP packets.

COLLISION TABLES

Current Value	Requested Value	Result
X	X	X
X	Y	If (X < Y) then X, else Y

Table I: Result of Second Link Timeout Request

Current Value	Requested Value	Result
N	0	N
N	N	N
N	M != N	Reject

Table II: Result of Second Flush Timeout Request