This addendum provides an optional update to the Bluetooth® Core Specification. When the addendum is applied to an allowed core specification, the following parts of the specification shall be replaced or appended with the revised versions according to Volume 1, Part B, Section 1.2:

Volume 0, Part B:
Bluetooth Compliance Requirements

Volume 1, Part D:
Mixing of Specification Versions

Volume 3, Part A:
Logical Link and Adaptation Protocol Specification
Master Table of Contents & Compliance Requirements

Covered Core Package versions:
2.0 + EDR
2.1 + EDR
Revision History

The Revision History is shown in the “Appendix” on page 41[vol. 0].

Contributors

The persons who contributed to this specification are listed in the Appendix.

Web Site

This specification can also be found on the official Bluetooth web site: http://www.bluetooth.com

Disclaimer and Copyright Notice

The copyright in these specifications is owned by the Promoter Members of Bluetooth SIG, Inc. (“Bluetooth SIG”). Use of these specifications and any related intellectual property (collectively, the “Specification”), is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the “Promoters Agreement”), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members (the “Membership Agreements”) and the Bluetooth Specification Early Adopters Agreements (“1.2 Early Adopters Agreements”) among Early Adopter members of the unincorporated Bluetooth special interest group and the Promoter Members (the “Early Adopters Agreement”). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a “Member”), is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.
THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth® technology (“Bluetooth® Products”) may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth® Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth® Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth® Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Copyright © 1999–2008

Ericsson AB,
Lenovo,
Intel Corporation,
Microsoft Corporation,
Motorola, Inc.,
Nokia Corporation,
Toshiba Corporation

*Third-party brands and names are the property of their respective owners.
This document specifies the requirements for Bluetooth compliance.
CONTENTS

BLUETOOTH COMPLIANCE REQUIREMENTS

1 Introduction ...........................................................................................................9
2 Scope ....................................................................................................................10
3 Definitions............................................................................................................11
   3.1 Types of Bluetooth Products ........................................................................11
4 Core Configurations.............................................................................................13
   4.1 Specification Naming Conventions ..............................................................13
   4.2 EDR Configurations .....................................................................................13
1 INTRODUCTION

The Bluetooth Qualification Program Reference Document (PRD) is the primary reference document for the Bluetooth Qualification Program and defines its requirements, functions, and policies. The PRD is available on the Bluetooth Web site.

Passing the Bluetooth Qualification Process demonstrates a certain measure of compliance and interoperability, but because products are not tested for every aspect of this Bluetooth Specification, qualification does not guarantee compliance. Passing the Bluetooth Qualification Process only satisfies one condition of the license grant. The Member has the ultimate responsibility to ensure that the qualified product complies with this Bluetooth Specification and interoperates with other products.
2 SCOPE

This part of the specification defines some fundamental concepts used in the Bluetooth Qualification Program.
3 DEFINITIONS

Bluetooth Qualification Process – The process defined in the Bluetooth Qualification Program Reference Document (PRD) to qualify a design used in implementations of Bluetooth wireless technology.

Bluetooth Qualification Program – The Bluetooth Qualification Process together with other related requirements and processes.

3.1 TYPES OF BLUETOOTH PRODUCTS

Bluetooth Product – Any product containing an implementation of Bluetooth wireless technology. All Bluetooth Products shall be one of the following:

- Bluetooth End Product
- Bluetooth Host Subsystem Product
- Bluetooth Controller Subsystem Product
- Bluetooth Profile Subsystem Product
- Bluetooth Component Product
- Bluetooth Development Tool
- Bluetooth Test Equipment

Bluetooth End Product - An implementation of Bluetooth wireless technology that implements, at a minimum, all mandatory requirements in Radio, Baseband, Link Manager, Logical Link Control and Adaptation Protocol, Service Discovery Protocol and Generic Access Profile parts of the Specification.

Bluetooth Subsystem Product - An implementation of Bluetooth wireless technology that implements only a portion of the Specification, in compliance with such portion of the Specification, and in accordance with the mandatory requirements as defined herein. Bluetooth Subsystem Products can be qualified solely for distribution and the use of Bluetooth wireless technology in Bluetooth Subsystem Products require such Bluetooth Subsystem Products to be combined with a complementary Bluetooth End Product or one or more complementary Bluetooth Subsystem Products such that the resulting combination satisfies the requirements of a Bluetooth End Product. There are three types of Bluetooth Subsystem Products as defined below:

- Bluetooth Host Subsystem Product – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in the Host Controller Interface, Logical Link Control and Adaptation Protocol, Service Discovery Protocol and Generic Access Profile parts of this Specification, but none of the protocols below Host Controller Interface (HCI). In addition, a Bluetooth Host Subsystem Product may contain, at a minimum, all the mandatory requirements defined in one or more of the protocols and profiles above HCI.
• **Bluetooth Controller Subsystem Product** – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in the Bluetooth Radio, Baseband, Link Manager and HCI parts of this Specification, but none of the Protocols and Profiles above HCI.

• **Bluetooth Profile Subsystem Product** – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in one or more of the profile specifications.

**Bluetooth Component Product** - An implementation of Bluetooth wireless technology, which does not meet the requirements of a Bluetooth End Product, but implements, at a minimum, all the mandatory requirements of either one or more of any of the protocol and profile parts of the Specification in compliance with such portion of the Specification. Bluetooth Component Products can be qualified solely for distribution and the use of the Bluetooth wireless technology in Bluetooth Component Products require such Bluetooth Component Products to be incorporated in Bluetooth End Products or Bluetooth Subsystem Products.

**Bluetooth Development Tool** - An implementation of Bluetooth wireless technology, intended to facilitate the development of new Bluetooth designs. Bluetooth Development Tools can be qualified solely for distribution and the use of the Bluetooth wireless technology in development of new Bluetooth Products.

**Bluetooth Test Equipment** - An implementation of Bluetooth wireless technology, intended to facilitate the testing of new Bluetooth Products. Bluetooth Test Equipment can be qualified solely for distribution and the use of the Bluetooth wireless technology in testing of new Bluetooth Products. Where necessary, Bluetooth Test Equipment may deviate from the Specification in order to fulfill the test purposes in the Bluetooth Test Specifications.
4  CORE CONFIGURATIONS

This section defines the set of features that are required for a product to be qualified to a specification name. Each core configuration is defined by a set of LMP feature bits or L2CAP feature bits that shall be supported to allow the configuration name to be used.

The configuration requirements imposed on a device depends on the profiles that it supports.

4.1  SPECIFICATION NAMING CONVENTIONS

Each specification is named by its core specification version number, followed by a list of the core configuration names that are implemented and qualified.

A complete specification name shall be stated as the core specification version number followed by “+”, and then either a single core configuration name or a sequence of core configuration names separated by “+”.

Examples of complete specification names including the core configuration names:

- Bluetooth v2.1
- Bluetooth v2.1 + EDR

In this example, a product claiming “Bluetooth v2.1” may implement some of the EDR features, following the requirements in other parts of the specifications, and be qualified for those features. If the applicable category(s) required in Section 4.2 for the incorporated profile(s) are not supported the “+ EDR” configuration name shall not be used in product literature.

4.2  EDR CONFIGURATIONS

This section specifies additional compliance requirements that shall be followed if the configuration name “EDR” is used within the complete specification name. The configuration name “EDR” may only be used with core specification version number 2.0 or later versions of the specification.

Table 4.1 defines three categories of Transport Requirements that shall be satisfied subject to the following rules:

- A Bluetooth product shall support category 1 whenever it supports asynchronous transports for the profiles it incorporates.
- A Bluetooth product shall support category 2 whenever it supports asynchronous transports with multislot ACL packets for the profiles it incorporates.
- A Bluetooth product shall support category 3 whenever it supports eSCO synchronous transports for the profiles it incorporates.
A multi-profile product shall support all applicable categories in order to use the "+ EDR" configuration name.

<table>
<thead>
<tr>
<th>Category No.</th>
<th>Transport Requirements</th>
<th>LMP Features Supported</th>
<th>L2CAP Feature Bits Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EDR for asynchronous transports (single slot)</td>
<td>Enhanced Data Rate ACL 2 Mbps mode (25) AND Enhanced Data Rate ACL 3 Mbps mode (26)</td>
<td>None</td>
</tr>
<tr>
<td>2</td>
<td>EDR for asynchronous transports (multi-slot)</td>
<td>3-slot Enhanced Data Rate ACL packets (39) AND 5-slot Enhanced Data Rate ACL packets (40)</td>
<td>None</td>
</tr>
<tr>
<td>3</td>
<td>EDR for synchronous transports</td>
<td>Enhanced Data Rate eSCO 2 Mbps mode (45)</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 4.1: EDR configuration requirements

Note: No additional requirements are stated on the support of 3-EV3, 2-EV5 and 3-EV5 packets.
Architecture & Terminology Overview

Covered Core Package versions:
2.0 + EDR
2.1 + EDR
Revision History

The Revision History is shown in the “Appendix” on page 41 [vol. 0].

Contributors

The persons who contributed to this specification are listed in the Appendix.

Web Site

This specification can also be found on the official Bluetooth web site:
http://www.bluetooth.com

Disclaimer and Copyright Notice

The copyright in these specifications is owned by the Promoter Members of Bluetooth SIG, Inc. (“Bluetooth SIG”). Use of these specifications and any related intellectual property (collectively, the “Specification”), is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the “Promoters Agreement”), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members (the “Membership Agreements”) and the Bluetooth Specification Early Adopters Agreements (“1.2 Early Adopters Agreements”) among Early Adopter members of the unincorporated Bluetooth special interest group and the Promoter Members (the “Early Adopters Agreement”). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a “Member”), is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.
THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth® technology (“Bluetooth® Products”) may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth® Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth® Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth® Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATE ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Copyright © 1999–2008

Ericsson AB,
Lenovo,
Intel Corporation,
Microsoft Corporation,
Motorola, Inc.,
Nokia Corporation,
Toshiba Corporation

*Third-party brands and names are the property of their respective owners.
MIXING OF SPECIFICATION VERSIONS
CONTENTS

1 Mixing of Specification Versions and Addendums .........................8
  1.1 Features and their Types ...................................................9
  1.2 Core Specification Addendums ..........................................10
1 MIXING OF SPECIFICATION VERSIONS AND ADDENDUMS

This part describes how different versions of the Core System Packages and adopted Addendums can be mixed in Bluetooth implementations. The Core System Packages consist of a Controller Package (see volume 2) and a Host Package (see volume 3). A Core Specification Addendum contains one or more Parts of the Core Specification. Addendums may be used to supersede a Part or add a Part to one or more Core Specifications according to the rules in section 1.2.

In order to describe how these packages can be mixed, one needs to distinguish between four categories of features specified in the different specification versions. The four categories are:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>Feature that exists below HCI and cannot be configured via HCI</td>
</tr>
<tr>
<td>Type 2</td>
<td>Feature that exists below HCI and can be configured/enabled via HCI</td>
</tr>
<tr>
<td>Type 3</td>
<td>Feature that exists below and above HCI and requires HCI command/events to function</td>
</tr>
<tr>
<td>Type 4</td>
<td>Feature that exists only above HCI</td>
</tr>
</tbody>
</table>

The outcome of mixing different core system packages are derived from the feature definitions in the table above:

- If an implementation contains features of type 1 or type 4, these features can function with any combination of Host Package and Controller Package versions.
- If an implementation contains features of type 2, these features can only be used under a default condition if a Host Package of an older version is mixed with a Controller Package of this version. In order to fully use the feature under all conditions, the Host Package and Controller Package must be of the same version.
- If an implementation contains features of type 3, these features can only function with a combination of Host Package and Controller Packages that support the feature.
1.1 FEATURES AND THEIR TYPES

The following table lists the features and their types.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Version</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic AFH operation</td>
<td>V1.2</td>
<td>1</td>
</tr>
<tr>
<td>Enhanced inquiry</td>
<td>V1.2</td>
<td>1</td>
</tr>
<tr>
<td>Configuration of AFH (setting channels and enabling/disabling channel assessment)</td>
<td>V1.2</td>
<td>2</td>
</tr>
<tr>
<td>Enhanced synchronization capability</td>
<td>V1.2</td>
<td>2</td>
</tr>
<tr>
<td>Interlaced inquiry scan</td>
<td>V1.2</td>
<td>2</td>
</tr>
<tr>
<td>Interlaced page scan</td>
<td>V1.2</td>
<td>2</td>
</tr>
<tr>
<td>Broadcast encryption</td>
<td>V1.2</td>
<td>2</td>
</tr>
<tr>
<td>Enhanced flow specification and flush time-out</td>
<td>V1.2</td>
<td>3</td>
</tr>
<tr>
<td>Extended SCO links</td>
<td>V1.2</td>
<td>3</td>
</tr>
<tr>
<td>Inquiry Result with RSSI</td>
<td>V1.2</td>
<td>3</td>
</tr>
<tr>
<td>L2CAP flow and error control</td>
<td>V1.2</td>
<td>4</td>
</tr>
<tr>
<td>2 Mbps EDR</td>
<td>V2.0 + EDR</td>
<td>2</td>
</tr>
<tr>
<td>3 Mbps EDR</td>
<td>V2.0 + EDR</td>
<td>2</td>
</tr>
<tr>
<td>3 slot packets in EDR</td>
<td>V2.0 + EDR</td>
<td>2</td>
</tr>
<tr>
<td>5 slot packets in EDR</td>
<td>V2.0 + EDR</td>
<td>2</td>
</tr>
<tr>
<td>2 Mbps eSCO</td>
<td>V2.0 + EDR</td>
<td>2*</td>
</tr>
<tr>
<td>3 Mbps eSCO</td>
<td>V2.0 + EDR</td>
<td>2*</td>
</tr>
<tr>
<td>3 slot packets for EDR eSCO</td>
<td>V2.0 + EDR</td>
<td>2*</td>
</tr>
<tr>
<td>Erroneous Data Reporting</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>Extended Inquiry Response</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>Encryption Pause and Resume</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>Link Supervision Timeout Changed Event</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>Non-Flushable Packet Boundary Flag</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>Sniff subrating</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>Secure Simple Pairing</td>
<td>V2.1 + EDR</td>
<td>3</td>
</tr>
<tr>
<td>L2CAP Enhanced Retransmission Mode</td>
<td>Addendum 1</td>
<td>4</td>
</tr>
<tr>
<td>L2CAP Streaming Mode</td>
<td>Addendum 1</td>
<td>4</td>
</tr>
</tbody>
</table>

The EDR eSCO options are marked as 2* because eSCO requires profile support, but if a product includes the eSCO option from V1.2, then EDR eSCO will be supported without any new support above HCI.
1.2 CORE SPECIFICATION ADDENDUMS

The purpose of a Core Specification Addendum (CSA) is to replace or add one or more Parts to a previously adopted Core Specification. Each Part within an Addendum is identified by a type indicating whether it may replace or add a part in an allowed Core Specification version.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replacement</td>
<td>The Part in the addendum is used instead of the equivalent part in the allowed Core Specification versions.</td>
</tr>
<tr>
<td>Addition</td>
<td>The Part in the Addendum is used in addition to the existing parts in the allowed Core Specification versions.</td>
</tr>
</tbody>
</table>

The following table contains a list of Core Specification Addendums and Core Specification versions with which they are allowed to be used.

<table>
<thead>
<tr>
<th>Addendum</th>
<th>Volume and Part</th>
<th>Type</th>
<th>Allowed Core Specification Versions</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Volume 0, Part B</td>
<td>Replacement</td>
<td>V2.0 + EDR, V2.1 + EDR</td>
<td>C.1</td>
</tr>
<tr>
<td>1</td>
<td>Volume 1, Part D</td>
<td>Replacement</td>
<td>V2.0 + EDR, V2.1 + EDR</td>
<td>Mandatory</td>
</tr>
<tr>
<td>1</td>
<td>Volume 3, Part A</td>
<td>Replacement</td>
<td>V2.0 + EDR, V2.1 + EDR</td>
<td>C.1</td>
</tr>
</tbody>
</table>

C1: Core Specificatin Addendum 1 requires at least one of these Parts to be supported.
Core System Package

[Host volume]

Covered Core Package versions:
2.0 + EDR
2.1 + EDR
Revision History

The Revision History is shown in the “Appendix” on page 41[vol. 0].

Contributors

The persons who contributed to this specification are listed in the “Appendix” on page 41[vol. 0].

Web Site

This specification can also be found on the official Bluetooth web site: http://www.bluetooth.com

Disclaimer and Copyright Notice

The copyright in these specifications is owned by the Promoter Members of Bluetooth SIG, Inc. (“Bluetooth SIG”). Use of these specifications and any related intellectual property (collectively, the “Specification”), is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the “Promoters Agreement”), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members (the “Membership Agreements”) and the Bluetooth Specification Early Adopters Agreements (“1.2 Early Adopters Agreements”) among Early Adopter members of the unincorporated Bluetooth special interest group and the Promoter Members (the “Early Adopters Agreement”). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a “Member”), is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.
THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth® technology (“Bluetooth® Products”) may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth® Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth® Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth® Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Copyright © 1999–2008

Ericsson AB,
Lenovo,
Intel Corporation,
Microsoft Corporation,
Motorola, Inc.,
Nokia Corporation,
Toshiba Corporation

*Third-party brands and names are the property of their respective owners.
The Bluetooth logical link control and adaptation protocol (L2CAP) supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. The protocol state machine, packet format, and composition are described in this document.
CONTENTS

Introduction ...................................................................................................11
  1.1 L2CAP Features ......................................................................................11
  1.2 Assumptions ............................................................................................15
  1.3 Scope .......................................................................................................15
  1.4 Terminology ............................................................................................16

2 General Operation .......................................................................................19
  2.1 Channel Identifiers ..................................................................................19
  2.2 Operation Between Devices ....................................................................19
  2.3 Operation Between Layers .....................................................................21
  2.4 Modes of Operation .................................................................................21

3 Data Packet Format .....................................................................................23
  3.1 Connection-oriented Channel in Basic L2CAP Mode .........................23
  3.2 Connectionless Data Channel in Basic L2CAP Mode .........................24
  3.3 Connection-oriented Channel in Retransmission/Flow Control/Streaming Modes .................................................................................25
     3.3.1 L2CAP header fields .........................................................................25
     3.3.2 Control field (2 octets) .....................................................................26
     3.3.3 L2CAP SDU length field (2 octets) ..................................................30
     3.3.4 Information payload field (0 to 65533 octets) ................................30
     3.3.5 Frame check sequence (2 octets) ....................................................31
     3.3.6 Invalid frame detection .....................................................................32
     3.3.7 Invalid Frame Detection Algorithm .................................................32

4 Signaling Packet Formats ..............................................................................34
  4.1 Command Reject (code 0x01) .................................................................36
  4.2 Connection Request (code 0x02) ..............................................................37
  4.3 Connection Response (code 0x03) ..........................................................39
  4.4 Configuration Request (code 0x04) ........................................................40
  4.5 Configuration Response (code 0x05) ......................................................43
  4.6 Disconnection Request (code 0x06) ........................................................45
  4.7 Disconnection Response (code 0x07) ......................................................46
  4.8 Echo Request (code 0x08) ......................................................................46
  4.9 Echo Response (code 0x09) ...................................................................47
  4.10 Information Request (code 0x0A) ..........................................................47
  4.11 Information Response (code 0x0B) .........................................................48
  4.12 Extended Feature Mask .........................................................................49

5 Configuration Parameter Options ...............................................................50
  5.1 Maximum Transmission Unit (MTU) ......................................................50
  5.2 Flush Timeout Option ..............................................................................52
5.3 Quality of Service (QoS) Option ........................................ 53
5.4 Retransmission and Flow Control Option ............................. 57
5.5 Frame Check Sequence (FCS) Option ................................ 62

6 State Machine ............................................................................. 64
6.1 General rules for the state machine: ....................................... 64
   6.1.1 CLOSED state ................................................................. 65
   6.1.2 WAIT_CONNECT_RSP state ........................................ 66
   6.1.3 WAIT_CONNECT state ..................................................... 66
   6.1.4 CONFIG state ................................................................. 67
   6.1.5 OPEN state .................................................................... 70
   6.1.6 WAIT_DISCONNECT state ............................................. 70
6.2 Timers events ........................................................................... 72
   6.2.1 RTX ............................................................................ 72
   6.2.2 ERTX ................................................................. 73

7 General Procedures ...................................................................... 76
7.1 Configuration Process ............................................................. 76
   7.1.1 Request path ................................................................. 77
   7.1.2 Response path ............................................................... 77
7.2 Fragmentation and Recombination ........................................... 78
   7.2.1 Fragmentation of L2CAP PDUs .................................... 78
   7.2.2 Recombination of L2CAP PDUs ................................... 79
7.3 Encapsulation of SDUs ............................................................. 80
   7.3.1 Segmentation of L2CAP SDUs ...................................... 80
   7.3.2 Reassembly of L2CAP SDUs ....................................... 81
   7.3.3 Segmentation and fragmentation .................................... 81
7.4 Delivery of Erroneous L2CAP SDUs ......................................... 82
7.5 Operation with Flushing ......................................................... 82
7.6 Connectionless Data Channel ................................................... 83

8 Procedures for Flow Control and Retransmission ....................... 84
8.1 Information Retrieval ............................................................. 84
8.2 Function of PDU Types for Flow Control and Retransmission... 84
   8.2.1 Information frame (I-frame) ......................................... 84
   8.2.2 Supervisory Frame (S-frame) ....................................... 84
      8.2.2.1 Receiver Ready (RR) ........................................... 84
      8.2.2.2 Reject (REJ) .................................................... 85
8.3 Variables and Sequence Numbers ........................................... 86
   8.3.1 Sending peer ............................................................... 86
      8.3.1.1 Send sequence number TxSeq ............................. 86
      8.3.1.2 Send state variable NextTXSeq ............................ 86
      8.3.1.3 Acknowledge state variable ExpectedAckSeq
8.3.2 Receiving peer ................................................................. 89
  8.3.2.1 Receive sequence number ReqSeq ......................... 89
  8.3.2.2 Receive state variable, ExpectedTxSeq ............... 89
  8.3.2.3 Buffer state variable BufferSeq .................... 89

8.4 Retransmission Mode ............................................................... 91
  8.4.1 Transmitting frames .................................................. 91
    8.4.1.1 Last received R was set to zero ................. 91
    8.4.1.2 Last received R was set to one ............... 93
  8.4.2 Receiving I-frames ................................................... 93
  8.4.3 I-frames pulled by the SDU reassembly function ........... 93
  8.4.4 Sending and receiving acknowledgements .................. 94
    8.4.4.1 Sending acknowledgements .................... 94
    8.4.4.2 Receiving acknowledgements ................ 94
  8.4.5 Receiving REJ frames ............................................... 95
  8.4.6 Waiting acknowledgements .................................. 95
  8.4.7 Exception conditions ........................................... 95
    8.4.7.1 TxSeq Sequence error .......................... 95
    8.4.7.2 ReqSeq Sequence error ......................... 96
    8.4.7.3 Timer recovery error .......................... 96
    8.4.7.4 Invalid frame .................................. 96

8.5 Flow Control Mode ............................................................... 97
  8.5.1 Transmitting I-frames ............................................ 97
  8.5.2 Receiving I-frames ................................................ 98
  8.5.3 I-frames pulled by the SDU reassembly function ........... 98
  8.5.4 Sending and receiving acknowledgements .................. 98
    8.5.4.1 Sending acknowledgements .................... 98
    8.5.4.2 Receiving acknowledgements ................ 98
  8.5.5 Waiting acknowledgements .................................. 99
  8.5.6 Exception conditions ........................................... 99
    8.5.6.1 TxSeq Sequence error .......................... 99
    8.5.6.2 ReqSeq Sequence error ......................... 100
    8.5.6.3 Invalid frame .................................. 100

8.6 Enhanced Retransmission Mode ............................................. 100
  8.6.1 Function Of PDU Types .......................................... 101
    8.6.1.1 Receiver Ready (RR) .......................... 101
    8.6.1.2 Reject (REJ) ...................................... 101
    8.6.1.3 Receiver Not Ready (RNR) ...................... 101
    8.6.1.4 Selective Reject (SREJ) ..................... 102
    8.6.1.5 Functions of the Poll (P) and Final (F) bits ... 102
  8.6.2 Rules For Timers .................................................. 102
  8.6.3 General Rules for the State Machine ....................... 103
  8.6.4 State Diagram ................................................... 104
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 Link Controller 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 upper layer data packets (L2CAP Service Data Units, SDU) up to 64 kilobytes in length. L2CAP also permits per-channel flow control and retransmission via the Flow Control and Retransmission Modes.

![Figure 1.1: L2CAP within protocol layers](image)

The L2CAP layer provides logical channels, named L2CAP channels, which are mapped to L2CAP logical links supported by an ACL logical transport, see baseband specification [vol.2, part B] Section 4.4 on page 92.

1.1 L2CAP FEATURES

The functional requirements for L2CAP include protocol/channel multiplexing, segmentation and reassembly (SAR), per-channel flow control, and error control. L2CAP lies above the Link Manager and Baseband and interfaces with higher layer protocols.

![Figure 1.2](image)

Figure 1.2 on page 12 breaks down L2CAP into its architectural components. The Channel Manager provides the control plane functionality and is responsible for all internal signaling, L2CAP peer-to-peer signaling and signaling with higher and lower layers. It performs the state machine functionality described in Section 6 on page 64 and uses message formats described in Section 4 on page 34, Section 5 on page 50. The Retransmission and Flow Control block provides per-channel flow control and optional retransmission for applications that require it. The Resource Manager is responsible for providing a frame relay service to the Channel Manager, the Retransmission and Flow Control...
block and those application data streams that do not require Retransmission and Flow Control services. It is responsible for coordinating the transmission and reception of packets related to multiple L2CAP channels over the facilities offered at the lower layer interface.

Figure 1.2: L2CAP architectural blocks
Protocol/channel multiplexing

L2CAP supports multiplexing because the Baseband Protocol does not support any ‘type’ field identifying the higher layer protocol being multiplexed above it.

During channel setup, protocol multiplexing capability is used to route the connection request to the correct upper layer protocol.

For data transfer, logical channel multiplexing is needed to distinguish between multiple upper layer entities. There may be more than one upper layer entity using the same protocol.

Segmentation and reassembly

With the frame relay service offered by the Resource Manager, the length of transport frames is controlled by the individual applications running over L2CAP. Many multiplexed applications are better served if L2CAP has control over the PDU length. This provides the following benefits:

a) Segmentation will allow the interleaving of application data units in order to satisfy latency requirements.

b) Memory and buffer management is easier when L2CAP controls the packet size.

c) Error correction by retransmission can be made more efficient.

d) The amount of data that is destroyed when an L2CAP PDU is corrupted or lost can be made smaller than the application’s data unit.

e) The application is decoupled from the segmentation required to map the application packets into the lower layer packets.

Flow control per L2CAP channel

When several data streams run over the same L2CAP logical link, using separate L2CAP channels, each channel may require individual flow control. Also L2CAP provides flow control services to profiles or applications that need flow control and can avoid having to implement it. Due to the delays between the L2CAP layers, stop-and-go flow control as employed in the baseband is not sufficient. A window based flow control scheme is provided. The use of flow control is an optional aspect of the L2CAP protocol.

Error control and retransmissions

Some applications require a residual error rate much smaller than the baseband can deliver. L2CAP includes optional error checks and retransmissions of L2CAP PDUs. The error checking in L2CAP protects against errors due to the baseband falsely accepting packet headers and due to failures of the HEC or CRC error checks on the baseband packets. Retransmission Mode also protects against loss of packets due to flush on the same logical transport. The error control works in conjunction with flow control in the sense that the flow control mechanism will throttle retransmissions as well as first transmissions. The use of error control and retransmission procedures is optional.
• Support for Streaming

Streaming applications such as audio set up an L2CAP channel with an agreed-upon data rate and do not want flow control mechanisms, including those in the baseband, to alter the flow of data. A flush timeout is used to keep data flowing on the transmitting side. Streaming mode can be used to stop baseband flow control from being applied on the receiving side.

• **Fragmentation and Recombination**

The lower layers have limited transmission capabilities and may require fragment sizes different from those created by L2CAP segmentation. Therefore layers below L2CAP may further fragment and recombine L2CAP PDUs to create fragments which fit each layer capabilities. During transmission of an L2CAP PDU, many different levels of fragmentation and recombination may occur in both peer devices.

The HCI driver or controller may fragment L2CAP PDUs to honor packet size constraints of a host controller interface transport scheme. This results in HCI data packet payloads carrying start and continuation fragments of the L2CAP PDU. Similarly the link controller may fragment L2CAP PDUs to map them into baseband packets. This results in baseband packet payloads carrying start and continuation fragments of the L2CAP PDU.

Each layer of the protocol stack may pass on different sized fragments of L2CAP PDUs, and the size of fragments created by a layer may be different in each peer device. However the PDU is fragmented within the stack, the receiving L2CAP entity still recombines the fragments to obtain the original L2CAP PDU.

• **Quality of Service**

The L2CAP connection establishment process allows the exchange of information regarding the quality of service (QoS) expected between two Bluetooth devices. Each L2CAP implementation monitors the resources used by the protocol and ensures that QoS contracts are honored.

L2CAP may support both isochronous and asynchronous data flows over the same L2CAP logical link by marking packets as automatically-flushable or non-automatically-flushable by setting the appropriate value for the Packet_Boundary_Flag in the HCI ACL Data Packet (see [vol.2, part E] Section 5.4.2 on page 394). Automatically-flushable L2CAP packets are flushed according to the automatic flush timeout set for the ACL logical transport on which the L2CAP logical link is mapped (see [vol.2, part E] Section 6.18 on page 407). Non-automatically-flushable L2CAP packets are not affected by the automatic flush timeout and will not be flushed. All L2CAP packets can be flushed by using the HCI Flush command (see [vol.2, part E] Section 7.3.4 on page 508).
1.2 ASSUMPTIONS

The protocol is designed based on the following assumptions:

1. The ACL logical transport and L2CAP logical link between two devices is set up using the Link Manager Protocol. The baseband provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than 1 unicast ACL logical transport 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. The L2CAP layer provides a channel with a degree of reliability based on the mechanisms available at the baseband layer and with optional additional packet segmentation and error detection that can be enabled in the enhanced L2CAP layer. The baseband performs data integrity checks 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 link controller protocol uses a 1-bit sequence number. 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.

4. Some applications will expect independent flow control, independence from the effects of other traffic and, in some cases, better error control than the baseband provides. The Flow and Error Control block provides two modes. Retransmission Mode offers segmentation, flow control and L2CAP PDU retransmissions. Flow control mode offers just the segmentation and flow control functions. If Basic L2CAP mode is chosen, the Flow and Error Control block is not used.

1.3 SCOPE

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

- L2CAP does not transport audio or transparent synchronous data designated for SCO or eSCO logical transports.
- L2CAP does not support a reliable multicast channel. See Section 3.2 on page 24.
- L2CAP does not support the concept of a global group name.
### 1.4 TERMINOLOGY

The following formal definitions apply:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper layer</td>
<td>The system layer above the L2CAP layer, which exchanges data with L2CAP in the form of SDUs. The upper layer may be represented by an application or higher protocol entity known as the Service Level Protocol. The interface of the L2CAP layer with the upper layer is not specified.</td>
</tr>
<tr>
<td>Lower layer</td>
<td>The system layer below the L2CAP layer, which exchanges data with the L2CAP layer in the form of PDUs, or fragments of PDUs. The lower layer is mainly represented within the Bluetooth Controller, however a Host Controller Interface (HCI) may be involved, such that an HCI host driver could also be seen as the lower layer. Except for the HCI functional specification (in case HCI is involved) the interface between L2CAP and the lower layer is not specified.</td>
</tr>
<tr>
<td>L2CAP channel</td>
<td>The logical connection between two endpoints in peer devices, characterized by their Channel Identifiers (CID), which is multiplexed on the L2CAP logical link, which is supported by an ACL logical transport, see [vol.2, part B] Section 4.4 on page 92</td>
</tr>
<tr>
<td>SDU, or L2CAP SDU</td>
<td>Service Data Unit: a packet of data that L2CAP exchanges with the upper layer and transports transparently over an L2CAP channel using the procedures specified here. The term SDU is associated with data originating from upper layer entities only, i.e. does not include any protocol information generated by L2CAP procedures.</td>
</tr>
<tr>
<td>Segment, or SDU segment</td>
<td>A part of an SDU, as resulting from the Segmentation procedure. An SDU may be split into one or more segments. Note: this term is relevant only to the Retransmission Mode and Flow Control Mode, not to the Basic L2CAP Mode.</td>
</tr>
<tr>
<td>Segmentation</td>
<td>A procedure used in the L2CAP Retransmission and Flow Control Modes, resulting in an SDU being split into one or more smaller units, called Segments, as appropriate for the transport over an L2CAP channel. Note: this term is relevant only to the Retransmission Mode and Flow Control Mode, not to the Basic L2CAP Mode.</td>
</tr>
<tr>
<td>Reassembly</td>
<td>The reverse procedure corresponding to Segmentation, resulting in an SDU being re-established from the segments received over an L2CAP channel, for use by the upper layer. Note that the interface between the L2CAP and the upper layer is not specified; therefore, reassembly may actually occur within an upper layer entity although it is conceptually part of the L2CAP layer. Note: this term is relevant only to the Retransmission Mode and Flow Control Mode, not to the Basic L2CAP Mode.</td>
</tr>
<tr>
<td>PDU, or L2CAP PDU</td>
<td>Protocol Data Unit a packet of data containing L2CAP protocol information fields, control information, and/or upper layer information data. A PDU is always started by a Basic L2CAP header. Types of PDUs are: B-frames, I-frames, S-frames, C-frames and G-frames.</td>
</tr>
</tbody>
</table>

*Table 1.1: Terminology*
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic L2CAP header</td>
<td>Minimum L2CAP protocol information that is present in the beginning of each PDU: a length field and a field containing the Channel Identifier (CID)</td>
</tr>
<tr>
<td>Basic information frame (B-frame)</td>
<td>A B-frame is a PDU used in the Basic L2CAP mode for L2CAP data packets. It contains a complete SDU as its payload, encapsulated by a basic L2CAP header.</td>
</tr>
<tr>
<td>Information frame (I-frame)</td>
<td>An I-frame is a PDU used in Enhanced Retransmission Mode, Streaming mode, Retransmission mode, and Flow Control Mode. It contains an SDU segment and additional protocol information, encapsulated by a basic L2CAP header.</td>
</tr>
<tr>
<td>Supervisory frame (S-frame)</td>
<td>An S-frame is a PDU used in Enhanced Retransmission Mode Retransmission mode, and Flow Control Mode. It contains protocol information only, encapsulated by a basic L2CAP header, and no SDU data.</td>
</tr>
<tr>
<td>Control frame (C-frame)</td>
<td>A C-frame is a PDU that contains L2CAP signaling messages exchanged between the peer L2CAP entities. C-frames are exclusively used on the L2CAP signaling channel.</td>
</tr>
<tr>
<td>Group frame (G-frame)</td>
<td>G-frame is a PDU exclusively used on Connectionless L2CAP channels in the Basic L2CAP mode. It contains a complete SDU as its payload, encapsulated by a specific header.</td>
</tr>
<tr>
<td>Fragment</td>
<td>A part of a PDU, as resulting from a fragmentation operation. Fragments are used only in the delivery of data to and from the lower layer. They are not used for peer-to-peer transportation. A fragment may be a Start or Continuation Fragment with respect to the L2CAP PDU. A fragment does not contain any protocol information beyond the PDU; the distinction of start and continuation fragments is transported by lower layer protocol provisions. Note: Start Fragments always begin with the Basic L2CAP header of a PDU.</td>
</tr>
<tr>
<td>Fragmentation</td>
<td>A procedure used to split L2CAP PDUs to smaller parts, named fragments, appropriate for delivery to the lower layer transport. Although described within the L2CAP layer, fragmentation may actually occur in an HCI host driver, and/or in the Controller, to accommodate the L2CAP PDU transport to HCI data packet or baseband packet sizes. Fragmentation of PDUs may be applied in all L2CAP modes. Note: in version 1.1, Fragmentation and Recombination was referred to as “Segmentation and Reassembly”.</td>
</tr>
<tr>
<td>Recombination</td>
<td>The reverse procedure corresponding to fragmentation, resulting in an L2CAP PDU re-established from fragments. In the receive path, full or partial recombination operations may occur in the Controller and/or the Host, and the location of recombination does not necessarily correspond to where fragmentations occurs on the transmit side.</td>
</tr>
<tr>
<td>Maximum Transmission Unit (MTU)</td>
<td>The maximum size of payload data, in octets, that the upper layer entity is capable of accepting, i.e. the MTU corresponds to the maximum SDU size.</td>
</tr>
</tbody>
</table>

*Table 1.1: Terminology*
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum PDU payload Size (MPS)</td>
<td>The maximum size of payload data in octets that the L2CAP layer entity is capable of accepting, i.e. the MPS corresponds to the maximum PDU payload size. Note: in the absence of segmentation, or in the Basic L2CAP Mode, the Maximum Transmission Unit is the equivalent to the Maximum PDU payload Size and shall be made equal in the configuration parameters.</td>
</tr>
<tr>
<td>Signaling MTU (MTU\textsubscript{sig})</td>
<td>The maximum size of command information that the L2CAP layer entity is capable of accepting. The MTU\textsubscript{sig}, refers to the signaling channel only and corresponds to the maximum size of a C-frame, excluding the Basic L2CAP header. The MTU\textsubscript{sig} value of a peer is discovered when a C-frame that is too large is rejected by the peer.</td>
</tr>
<tr>
<td>Connectionless MTU (MTU\textsubscript{cnl})</td>
<td>The maximum size of the connection packet information that the L2CAP layer entity is capable of accepting. The MTU\textsubscript{cnl}, refers to the connectionless channel only and corresponds to the maximum G-frame, excluding the Basic L2CAP header. The MTU\textsubscript{cnl} of a peer can be discovered by sending an Information Request.</td>
</tr>
<tr>
<td>MaxTransmit</td>
<td>In Retransmission mode, MaxTransmit controls the number of transmissions of a PDU that L2CAP is allowed to try before assuming that the PDU (and the link) is lost. The minimum value is 1 (only 1 transmission permitted). Note: Setting MaxTransmit to 1 prohibits PDU retransmissions. Failure of a single PDU will cause the link to drop. By comparison, in Flow Control mode, failure of a single PDU will not necessarily cause the link to drop.</td>
</tr>
</tbody>
</table>

Table 1.1: Terminology
2 GENERAL OPERATION

L2CAP is based around the concept of 'channels'. Each one of the endpoints of an L2CAP channel is referred to by a channel identifier (CID).

2.1 CHANNEL IDENTIFIERS

A channel identifier (CID) is the local name representing a logical channel endpoint on the device. The null identifier (0x0000) is an illegal identifier and shall never be used as a destination endpoint. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that two simultaneously active L2CAP channels shall not share the same CID. Table 2.1 on page 19 summarizes 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.

<table>
<thead>
<tr>
<th>CID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Null identifier</td>
</tr>
<tr>
<td>0x0001</td>
<td>Signaling channel</td>
</tr>
<tr>
<td>0x0002</td>
<td>Connectionless reception channel</td>
</tr>
<tr>
<td>0x0003-0x003F</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0040-0xFFFF</td>
<td>Dynamically allocated</td>
</tr>
</tbody>
</table>

Table 2.1: CID name space

2.2 OPERATION BETWEEN DEVICES

Figure 2.1 on page 20 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 signaling 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 connection oriented and connectionless channels. Support for a signaling channel within an L2CAP entity is mandatory.
Note: it is assumed that an L2CAP signaling channel is available immediately when an ACL logical transport is established between two devices, and L2CAP traffic is enabled on the L2CAP logical link. 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](image)

Table 2.2 on page 20 describes the various channels and their source and destination identifiers. A CID is allocated to identify the local endpoint and shall be in the range 0x0040 to 0xFFFF. Section 6 on page 64 describes the state machine associated with each connection-oriented channel. Section 3.1 on page 23 describes the packet format associated with bi-directional channels and Section 3.2 on page 24 describes the packet format associated with unidirectional channels.

<table>
<thead>
<tr>
<th>Channel Type</th>
<th>Local CID (sending)</th>
<th>Remote CID (receiving)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection-oriented</td>
<td>Dynamically allocated</td>
<td>Dynamically allocated</td>
</tr>
<tr>
<td>Connectionless data</td>
<td>Dynamically allocated</td>
<td>0x0002 (fixed)</td>
</tr>
<tr>
<td>Signaling</td>
<td>0x0001 (fixed)</td>
<td>0x0001 (fixed)</td>
</tr>
</tbody>
</table>

Table 2.2: Types of Channel Identifiers
2.3 OPERATION BETWEEN LAYERS

L2CAP implementations should follow the general architecture described below. L2CAP implementations transfer data between upper layer protocols and the lower layer protocol. This document lists a number of services that should be exported by any L2CAP implementation. Each implementation shall also support a set of signaling 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 implementation specific.

![L2CAP transaction model](image)

2.4 MODES OF OPERATION

L2CAP may operate in one of five different modes as selected for each L2CAP channel by an upper layer.

The modes are:

- Basic L2CAP Mode (equivalent to L2CAP specification in Bluetooth v1.1) ¹
- Flow Control Mode
- Retransmission Mode
- Enhanced Retransmission Mode
- Streaming Mode

The modes are enabled using the configuration procedure described in Section 7.1. The Basic L2CAP Mode shall be the default mode, which is used when no other mode is agreed. Enhanced Retransmission mode should be enabled for reliable channels. Streaming mode should be enabled for streaming applications. Either Enhanced Retransmission mode or Streaming mode should be enabled when supported by both L2CAP entities. Flow Control Mode and

---

Retransmission mode shall only be enabled when communicating with L2CAP entities that do not support either Enhanced Retransmission mode or Streaming mode.

In Flow Control mode Retransmission mode, and Enhanced Retransmission mode, PDUs exchanged with a peer entity are numbered and acknowledged. The sequence numbers in the PDUs are used to control buffering, and a TxWindow size is used to limit the required buffer space and/or provide a method for flow control.

In Flow Control Mode no retransmissions take place, but missing PDUs are detected and can be reported as lost.

In Retransmission Mode a timer is used to ensure that all PDUs are delivered to the peer, by retransmitting PDUs as needed. A go-back-n repeat mechanism is used to simplify the protocol and limit the buffering requirements.

Enhanced Retransmission mode is similar to Retransmission mode. It adds the ability to set a POLL bit to solicit a response from a remote L2CAP entity, adds the SREJ S-frame to improve the efficiency of error recovery and adds an RNR S-frame to replace the R-bit for reporting a local busy condition.

Streaming mode is for real-time isochronous traffic. PDUs are numbered but are not acknowledged. A finite flush timeout is set on the sending side to flush packets that are not sent in a timely manner. On the receiving side if the receive buffers are full when a new PDU is received then a previously received PDU is overwritten by the newly received PDU. This is a method of flow control. Missing PDUs can be detected and reported as lost. TxWindow size is not used in Streaming mode.
3 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. All packet fields shall use Little Endian byte order.

3.1 CONNECTION-ORIENTED CHANNEL IN BASIC L2CAP MODE

Figure 3.1 on page 23 illustrates the format of the L2CAP PDU within a connection-oriented channel. In basic L2CAP mode, the L2CAP PDU on a connection-oriented channel is also referred to as a "B-frame".

![Figure 3.1: PDU format in Basic L2CAP mode on connection-oriented channels (field sizes in bits)](image)

The fields shown are:

- **Length**: 2 octets (16 bits)
  Length indicates the size of the information payload in octets, excluding the length of the L2CAP header. The length of an information payload can be up to 65535 octets. The Length field is used for recombination and serves as a simple integrity check of the recombined L2CAP packet on the receiving end.

- **Channel ID**: 2 octets
  The channel ID (CID) identifies the destination channel endpoint of the packet.

- **Information payload**: 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 MTU is determined during channel configuration (see Section 5.1 on page 50). The minimum supported MTU for the signaling PDUs ($MTU_{\text{sig}}$) is 48 octets (see Section 4 on page 34).
3.2 CONNECTIONLESS DATA CHANNEL IN BASIC L2CAP MODE

Figure 3.2 illustrates the L2CAP PDU format within a connectionless data channel. Here, the L2CAP PDU is also referred to as a "G-frame".

The fields shown are:

- **Length**: 2 octets
  Length indicates the size of information payload plus the PSM field in octets.

- **Channel ID**: 2 octets
  Channel ID (0x0002) reserved for connectionless traffic.

- **Protocol/Service Multiplexer (PSM)**: 2 octets (minimum)
  For information on the PSM field see Section 4.2 on page 37.

- **Information payload**: 0 to 65533 octets
  The payload information to be distributed to all members of the piconet. Implementations shall support a connectionless MTU (MTU_{cnl}) of 48 octets on connectionless channels. Devices may also explicitly change to a larger or smaller connectionless MTU (MTU_{cnl}). Note: the maximum size of the Information payload field decreases accordingly if the PSM field is extended beyond the two octet minimum.
3.3 CONNECTION-ORIENTED CHANNEL IN RETRANSMISSION/FLOW CONTROL/STREAMING MODES

To support flow control and retransmissions, L2CAP PDU types with protocol elements in addition to the Basic L2CAP header are defined. The information frames (I-frames) are used for information transfer between L2CAP entities. The supervisory frames (S-frames) are used to acknowledge I-frames and request retransmission of I-frames.

![L2CAP PDU formats in Flow Control and Retransmission Modes](image)

3.3.1 L2CAP header fields

- **Length**: 2 octets
  
The first two octets in the L2CAP PDU contain the length of the entire L2CAP PDU in octets, excluding the Length and CID field.

  For I-frames and S-frames, the Length field therefore includes the octet lengths of the Control, L2CAP SDU Length (when present), Information octets and frame check sequence (FCS) (when present) fields.

  The maximum number of Information octets in one I-frame is based on which fields are present as follows:

  - L2CAP SDU Length present and FCS present: 65529 octets
  - L2CAP SDU Length present and FCS not present: 65531 octets
  - L2CAP SDU Length not present and FCS present: 65531 octets
  - L2CAP SDU Length not present and FCS not present: 65533 octets

- **Channel ID**: 2 octets
  
  This field contains the Channel Identification (CID).
3.3.2 Control field (2 octets)

The Control Field identifies the type of frame. There are two different Control Field formats: the Standard Control Field and the Enhanced Control Field. The Standard Control Field shall be used for Retransmission mode and Flow Control mode. The Enhanced Control Field shall be used for Enhanced Retransmission mode and Streaming mode. The Control Field will contain sequence numbers where applicable. Its coding is shown in Table 3.1 on page 26 and Table 3.2 on page 27. There are two different frame types, Information frame types and Supervisory frame types. Information and Supervisory frames types are distinguished by the rightmost bit in the Control field, as shown in Table 3.1 on page 26 and Table 3.2 on page 27.

- **Information frame format (I-frame)**
  
The I-frames are used to transfer information between L2CAP entities. Each I-frame has a TxSeq (Send sequence number), ReqSeq (Receive sequence number) which may or may not acknowledge additional I-frames received by the data link layer entity. Each I-frame with a Standard Control field has a retransmission bit (R bit) that affects whether I-frames are retransmitted. Each I-frame with an Enhanced Control Field has an F-bit that is used in Poll/Final bit functions.

  The SAR field in the I-frame is used for segmentation and reassembly control. The L2CAP SDU Length field specifies the length of an SDU, including the aggregate length across all segments if segmented.

- **Supervisory frame format (S-frame)**
  
The S-frames are used to acknowledge I-frames and request retransmission of I-frames. Each S-frame has an ReqSeq sequence number which may acknowledge additional I-frames received by the data link layer entity. Each S-frame with a Standard Control Field has a retransmission bit (R bit) that affects whether I-frames are retransmitted. Each S-frame with an Enhanced Control field has a Poll bit (P-bit) and a Final bit (F-bit) and does not have an R-bit.

  Defined types of S-frames are RR (Receiver Ready), REJ (Reject), RNR (Receiver Not Ready) and SREJ (Selective Reject).

<table>
<thead>
<tr>
<th>Frame type</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>TxSeq</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>X</td>
<td>X</td>
<td>ReqSeq</td>
<td>R</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>S</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X denotes reserved bits. Shall be coded 0.

*Table 3.1: Standard Control Field formats*

- **Send Sequence Number - TxSeq (6 bits)**
  
The send sequence number is used to number each I-frame, to enable sequencing and retransmission.
The receive sequence number is used by the receiver side to acknowledge I-frames, and in the REJ and SREJ frames to request the retransmission of an I-frame with a specific send sequence number.

Table 3.2: Enhanced Control Field formats

- **Receive Sequence Number - ReqSeq (6 bits)**
  The receive sequence number is used by the receiver side to acknowledge I-frames, and in the REJ and SREJ frames to request the retransmission of an I-frame with a specific send sequence number.

<table>
<thead>
<tr>
<th>Frame type</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X denotes reserved bits. Shall be coded 0.
• **Retransmission Disable Bit - R (1 bit)**
  The Retransmission Disable bit is used to implement Flow Control. The receiver sets the bit when its internal receive buffer is full, this happens when one or more I-frames have been received but the SDU reassembly function has not yet pulled all the frames received. When the sender receives a frame with the Retransmission Disable bit set it shall disable the RetransmissionTimer, this causes the sender to stop retransmitting I-frames.
  
  R=0: Normal operation. Sender uses the RetransmissionTimer to control retransmission of I-frames. Sender does not use the MonitorTimer.
  
  R=1: Receiver side requests sender to postpone retransmission of I-frames. Sender monitors signaling with the MonitorTimer. Sender does not use the RetransmissionTimer.

  The functions of ReqSeq and R are independent.

• **Segmentation and Reassembly - SAR (2 bits)**
  The SAR bits define whether an L2CAP SDU is segmented. For segmented SDUs, the SAR bits also define which part of an SDU is in this I-frame, thus allowing one L2CAP SDU to span several I-frames.
  
  An I-frame with SAR="Start of L2CAP SDU" also contains a length indicator, specifying the number of information octets in the total L2CAP SDU. The encoding of the SAR bits is shown in Table 3.3.

<table>
<thead>
<tr>
<th>SAR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Unsegmented L2CAP SDU</td>
</tr>
<tr>
<td>01</td>
<td>Start of L2CAP SDU</td>
</tr>
<tr>
<td>10</td>
<td>End of L2CAP SDU</td>
</tr>
<tr>
<td>11</td>
<td>Continuation of L2CAP SDU</td>
</tr>
</tbody>
</table>

*Table 3.3: SAR control element format.*

• **Supervisory function - S (2 bits)**
  The S-bits mark the type of S-frame. There are four types defined: RR (Receiver Ready), REJ (Reject), RNR (Receiver Not Ready) and SREJ (Selective Reject). The encoding is shown in Table 3.4.

<table>
<thead>
<tr>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>RR - Receiver Ready</td>
</tr>
<tr>
<td>01</td>
<td>REJ - Reject</td>
</tr>
<tr>
<td>10</td>
<td>RNR - Receiver Not Ready</td>
</tr>
<tr>
<td>11</td>
<td>SREJ - Select Reject</td>
</tr>
</tbody>
</table>

*Table 3.4: S control element format: type of S-frame.*

• **Poll - P (1 bit)**
  The P-bit is set to 1 to solicit a response from the receiver. The receiver shall respond immediately with a frame with the F-bit set to 1.

• **Final - F (1 bit)**
The F-bit is set to 1 in response to an S-frame with the P bit set to 1.
3.3.3 L2CAP SDU length field (2 octets)

When a SDU spans more than one I-frame, the first I-frame in the sequence shall be identified by SAR=01="Start of L2CAP SDU". The L2CAP SDU Length field shall specify the total number of octets in the SDU. The L2CAP SDU Length field shall be present in I-frames with SAR=01 (Start of L2CAP SDU), and shall not be used in any other I-frames. When the SDU is unsegmented (SAR=00), L2CAP SDU Length field is not needed and shall not be present.

3.3.4 Information payload field (0 to 65533 octets)

The information payload field consists of an integral number of octets. The maximum number of octets in this field is the same as the negotiated value of the MPS configuration parameter. The maximum number of octets in this field is also limited by the range of the basic L2CAP header length field. This ranges from 65533 octets for I-frames without an SDU length field and an FCS field to 65529 octets for I-frames with both an SDU length field and FCS field. Thus, even if an MPS of 65533 has been negotiated, the range of the basic L2CAP header length field will restrict the number of octets in this field. For example, when both an SDU length field and FCS field are present the number of octets in this field is restricted to 65529.
3.3.5 Frame check sequence (2 octets)

The Frame Check Sequence (FCS) is 2 octets. The FCS is constructed using the generator polynomial \( g(D) = D^{16} + D^{15} + D^2 + 1 \) (see Figure 3.4). The 16 bit LFSR is initially loaded with the value 0x0000, as depicted in Figure 3.5. The switch S is set in position 1 while data is shifted in, LSB first for each octet. After the last bit has entered the LFSR, the switch is set in position 2, and the register contents are transmitted from right to left (i.e. starting with position 15, then position 14, etc.). The FCS covers the Basic L2CAP header, Control, L2CAP-SDU length and Information payload fields, if present, as shown in Figure 3.3 on page 25.

![Figure 3.4: The LFSR circuit generating the FCS.](image)

Examples of FCS calculation, \( g(D) = D^{16} + D^{15} + D^2 + 1 \):

1. **I Frame**
   
   Length = 14
   Control = 0x0002 (SAR=0, ReqSeq=0, R=0, TxSeq=1)
   Information Payload = 00 01 02 03 04 05 06 07 08 09 (10 octets, hexadecimal notation)
   ==> FCS = 0x6138
   ==> Data to Send = 0E 00 40 00 02 00 01 02 03 04 05 06 07 08 09 38 61 (hexadecimal notation)

2. **RR Frame**
   
   Length = 4
   Control = 0x0101 (ReqSeq=1, R=0, S=0)
   ==> FCS = 0x14D4
   ==> Data to Send = 04 00 40 00 01 01 D4 14 (hexadecimal notation)
3.3.6 Invalid frame detection

For Retransmission mode and Flow Control mode, a received PDU shall be regarded as invalid, if one of the following conditions occurs:

1. Contains an unknown CID.
2. Contains an FCS error.
3. Contains a length greater than the maximum PDU payload size (MPS).
4. I-frame that has fewer than 8 octets.
5. I-frame with SAR=01 (Start of L2CAP SDU) that has fewer than 10 octets.
6. I-frame with SAR bits that do not correspond to a normal sequence of either unsegmented or start, continuation, end for the given CID.
7. S-frame where the length field is not equal to 4.

These error conditions may be used for error reporting.

3.3.7 Invalid Frame Detection Algorithm

For Enhanced Retransmission mode and Streaming mode the following algorithm shall be used for received PDUs. It may be used for Retransmission mode and Flow Control mode:

1. Check the CID. If the PDU contains an unknown CID then it shall be ignored.
2. Check the FCS. If the PDU contains an FCS error then it shall be ignored. If the channel is configured to use "No FCS" then the PDU is considered to have a good FCS (no FCS error).
3. Check the following conditions. If one of the conditions occurs the channel shall be closed or in the case of fixed channels the ACL shall be disconnected.
   a) PDU contains a length greater than the maximum PDU payload size (MPS)
   b) I-frame that has fewer than the required number of octets. If the channel is configured to use "No FCS" then the required number of octets is 6 otherwise the required number of octets is 8.
   c) S-frame where the length field is invalid. If the channel is configured to use "No FCS" then the length field shall be 2 otherwise the length field shall be 4.
4. Check the SAR bits. The SAR check is performed after the frame has been successfully received in the correct sequence. The PDU is invalid if one of the following conditions occurs:
   a) I-frame with SAR=01 (Start of L2CAP SDU) that has fewer than the required number of octets. If the channel is configured to use
"No FCS" then the required number of octets is 8 otherwise the required number of octets is 10.

b) I-frame with SAR bits that do not correspond to a normal sequence of either unsegmented or start, continuation, end for the given CID.

c) I-frame with SAR= 01 (Start of L2CAP SDU) where the value in the L2CAP SDU length field exceeds the configured MTU.

5. If the I-frame has been received in the correct sequence and is invalid as described in 4 then the channel shall be closed or in the case of fixed channels the ACL shall be disconnected. For Streaming mode and Flow Control mode if one or more I-frames are missing from a sequence of I-frames using SAR bits of start, continuation and end then received I-frames in the sequence may be ignored. For Flow Control mode and Streaming mode I-frames received out of sequence with SAR bits of unsegmented may be accepted.

If the algorithm is used for Retransmission mode or Flow control mode then it shall be used instead of Invalid Frame detection described in section 3.3.6.

These error conditions may be used for error reporting.
4 SIGNALING PACKET FORMATS

This section describes the signaling commands passed between two L2CAP entities on peer devices. All signaling commands are sent to the signaling channel with CID 0x0001. This signaling channel is available as soon as an ACL logical transport is set up and L2CAP traffic is enabled on the L2CAP logical link. Figure 4.1 on page 34 illustrates the general format of L2CAP PDUs containing signaling commands (C-frames). Multiple commands may be sent in a single C-frame. Commands take the form of Requests and Responses. All L2CAP implementations shall support the reception of C-frames with a payload length that does not exceed the signaling MTU. The minimum supported payload length for the C-frame ($MTU_{\text{sig}}$) is 48 octets. L2CAP implementations should not use C-frames that exceed the $MTU_{\text{sig}}$ of the peer device. If they ever do, the peer device shall send a Command Reject containing the supported $MTU_{\text{sig}}$. Implementations must be able to handle the reception of multiple commands in an L2CAP packet.

![Figure 4.1: L2CAP PDU format on the signaling channel](image)

Figure 4.2 displays the general format of all signaling commands.

![Figure 4.2: Command format](image)
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 4.1 on page 36) is sent in response.
  
  Table 4.1 on page 35 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RESERVED</td>
</tr>
<tr>
<td>0x01</td>
<td>Command reject</td>
</tr>
<tr>
<td>0x02</td>
<td>Connection request</td>
</tr>
<tr>
<td>0x03</td>
<td>Connection response</td>
</tr>
<tr>
<td>0x04</td>
<td>Configure request</td>
</tr>
<tr>
<td>0x05</td>
<td>Configure response</td>
</tr>
<tr>
<td>0x06</td>
<td>Disconnection request</td>
</tr>
<tr>
<td>0x07</td>
<td>Disconnection response</td>
</tr>
<tr>
<td>0x08</td>
<td>Echo request</td>
</tr>
<tr>
<td>0x09</td>
<td>Echo response</td>
</tr>
<tr>
<td>0x0A</td>
<td>Information request</td>
</tr>
<tr>
<td>0x0B</td>
<td>Information response</td>
</tr>
</tbody>
</table>

Table 4.1: Signaling Command Codes

- **Identifier (1 octet)**
  The Identifier field is one octet long and matches responses with requests. The requesting device sets this field and the responding device uses the same value in its response. Between any two devices a different Identifier shall be used for each successive command. Following the original transmission of an Identifier in a command, the Identifier may be recycled if all other Identifiers have subsequently been used.

  RTX and ERTX timers are used to determine when the remote end point is not responding to signaling requests. On the expiration of a RTX or ERTX timer, the same identifier shall be used if a duplicate Request is re-sent as stated in Section 6.2 on page 72.

  A device receiving a duplicate request should reply with a duplicate response. A command response with an invalid identifier is silently discarded. Signaling identifier 0x00 is an illegal identifier and shall never be used in any command.
• **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. The Code field determines the format of the Data field. The Length field determines the length of the data field.

### 4.1 COMMAND REJECT (CODE 0x01)

A Command Reject packet shall be sent in response to a command packet with an unknown command code or when sending the corresponding response is inappropriate. Figure 4.3 on page 36 displays the format of the packet. The identifier shall match the identifier of the command packet being rejected. Implementations shall always send these packets in response to unidentified signaling packets. Command Reject packets should not be sent in response to an identified Response packet.

When multiple commands are included in an L2CAP packet and the packet exceeds the signaling MTU (MTU_sig) of the receiver, a single Command Reject packet shall be sent in response. The identifier shall match the first Request command in the L2CAP packet. If only Responses are recognized, the packet shall be silently discarded.

![Figure 4.3: Command Reject packet](image)

**Figure 4.3** shows the format of the Command Reject packet. The data fields are:

• **Reason (2 octets)**
  The Reason field describes why the Request packet was rejected, and is set to one of the Reason codes in Table 4.2.

<table>
<thead>
<tr>
<th>Reason value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Command not understood</td>
</tr>
<tr>
<td>0x0001</td>
<td>Signaling MTU exceeded</td>
</tr>
<tr>
<td>0x0002</td>
<td>Invalid CID in request</td>
</tr>
<tr>
<td>Other</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Table 4.2: 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, “Signaling MTU Exceeded”, the 2-octet Data field represents the maximum signaling 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. The data field shall be 4 octets containing the local (first) and remote (second) channel endpoints (relative to the sender of the Command Reject) of the disputed channel. The remote endpoint is the source CID from the rejected command. The local endpoint is the destination CID from the rejected command. If the rejected command contains only one of the channel endpoints, the other one shall be replaced by the null CID 0x0000.

<table>
<thead>
<tr>
<th>Reason value</th>
<th>Data Length</th>
<th>Data value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0 octets</td>
<td>N/A</td>
</tr>
<tr>
<td>0x0001</td>
<td>2 octets</td>
<td>Actual MTU&lt;sub&gt;sig&lt;/sub&gt;</td>
</tr>
<tr>
<td>0x0002</td>
<td>4 octets</td>
<td>Requested CID</td>
</tr>
</tbody>
</table>

Table 4.3: Reason Data values

4.2 CONNECTION REQUEST (CODE 0x02)

Connection request packets are sent to create an L2CAP channel between two devices. The L2CAP channel shall be established before configuration begins. Figure 4.4 illustrates a Connection Request packet.

Figure 4.4: Connection Request Packet
The data fields are:

• **Protocol/Service Multiplexer - PSM (2 octets (minimum))**
  
  The PSM field is at least two octets in length. The structure of the PSM field is based on the ISO 3309 extension mechanism for address fields. All PSM values shall be ODD, that is, the least significant bit of the least significant octet must be ‘1’. Also, all PSM values shall have the least significant bit of the most significant octet equal to ‘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.

  PSM values are defined in the [Assigned Numbers](#) document.

• **Source CID - SCID (2 octets)**
  
  The source 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 to the sender of the request shall be sent to this CID. Thus, the Source CID represents the channel endpoint on the device sending the request and receiving the response.
4.3 CONNECTION RESPONSE (CODE 0x03)

When a device receives a Connection Request packet, it shall send a Connection Response packet. The format of the connection response packet is shown in Figure 4.5.

![Figure 4.5: Connection Response Packet](image)

The data fields are:

- **Destination Channel Identifier - DCID (2 octets)**
  
  This field contains the channel endpoint on the device sending this Response packet. Thus, the Destination CID represents the channel endpoint on the device receiving the request and sending the response.

- **Source Channel Identifier - SCID (2 octets)**
  
  This field contains the channel endpoint on the device receiving this Response packet. This is copied from the SCID field of the connection request 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 or is pending. A logical channel is established on the receipt of a successful result. Table 4.4 on page 39 defines values for this field. The DCID and SCID fields shall be ignored when the result field indicates the connection was refused.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Connection successful.</td>
</tr>
<tr>
<td>0x0001</td>
<td>Connection pending</td>
</tr>
<tr>
<td>0x0002</td>
<td>Connection refused – PSM not supported.</td>
</tr>
<tr>
<td>0x0003</td>
<td>Connection refused – security block.</td>
</tr>
<tr>
<td>0x0004</td>
<td>Connection refused – no resources available.</td>
</tr>
<tr>
<td>Other</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

*Table 4.4: Result values*
• Status (2 octets)

Only defined for Result = Pending. Indicates the status of the connection. The status is set to one of the values shown in Table 4.5 on page 40.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>No further information available</td>
</tr>
<tr>
<td>0x0001</td>
<td>Authentication pending</td>
</tr>
<tr>
<td>0x0002</td>
<td>Authorization pending</td>
</tr>
<tr>
<td>Other</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 4.5: Status values

4.4 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 shall be related exclusively to either the outgoing or the incoming data traffic but not both of them. In Section 5 on page 50, the various configuration parameters and their relation to the outgoing or incoming data traffic are shown. If an L2CAP entity receives a Configuration Request while it is waiting for a response it shall not block sending the Configuration Response, otherwise the configuration process may deadlock.

If no parameters need to be negotiated then no options shall be inserted and the continuation flag (C) shall be set to zero. L2CAP entities in remote devices shall 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 explicitly or implicitly accepted values. Even if all default values are acceptable, a Configuration Request packet with no options shall be sent. Implicitly accepted values are default values for the configuration parameters that have not been explicitly negotiated for the specific channel under configuration.

Each configuration parameter is one-directional. The configuration parameters describe the non default parameters the device sending the Configuration Request will accept. The configuration request can not request a change in the parameters the device receiving the request will accept.

If a device needs to establish the value of a configuration parameter the remote device will accept, then it must wait for a configuration request containing that configuration parameter to be sent from the remote device.

See Section 7.1 on page 76 for details of the configuration procedure.
Figure 4.6 defines the format of the Configuration Request packet.

<table>
<thead>
<tr>
<th>LSB</th>
<th>octet 0</th>
<th>octet 1</th>
<th>octet 2</th>
<th>octet 3</th>
<th>MSB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code=0x04</td>
<td>Identifier</td>
<td>Length</td>
<td>Destination CID</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Configuration Options</td>
</tr>
</tbody>
</table>

**Figure 4.6: Configuration Request Packet**

The data fields are:

- **Destination CID - DCID (2 octets)**
  
  This field contains the channel endpoint on the device receiving this Request packet.

- **Flags (2 octets)**
  
  Figure 4.7 shows the two-octet Flags field. Note the most significant bit is shown on the left.

**Figure 4.7: Configuration Request Flags field format**

Only one flag is defined, the Continuation flag (C).

When all configuration options cannot fit into a Configuration Request with length that does not exceed the receiver's MTU_{sig}, the options shall be passed in multiple configuration command packets. If all options fit into the receiver's MTU_{sig}, then they shall be sent in a single configuration request with the continuation flag set to zero. Each Configuration Request shall contain an integral number of options - partially formed options shall not be sent in a packet. Each Request shall be tagged with a different Identifier and shall be matched with a Response with the same Identifier.

When used in the Configuration Request, the continuation flag indicates the responder should expect to receive multiple request packets. The responder shall reply to each Configuration Request packet. The responder may reply to each Configuration Request with a Configuration Response containing the same option(s) present in the Request (except for those error conditions more appropriate for a Command Reject), or the responder may reply with a "Success" Configuration Response packet containing no options, delaying those options until the full Request has been received. The Configuration Request packet with the continuation flag cleared shall be treated as the Configuration Request event in the channel state machine.
When used in the Configuration Response, the continuation flag shall be set to one if the flag is set to one in the Request. If the continuation flag is set to one in the Response when the matching Request has the flag set to zero, it indicates the responder has additional options to send to the requestor. In this situation, the requestor shall send null-option Configuration Requests (with continuation flag set to zero) to the responder until the responder replies with a Configuration Response where the continuation flag is set to zero. The Configuration Response packet with the continuation flag set to zero shall be treated as the Configuration Response event in the channel state machine.

The result of the configuration transaction is the union of all the result values. All the result values must succeed for the configuration transaction to succeed.

Other flags are reserved and shall be set to zero. L2CAP implementations shall ignore these bits.

- **Configuration Options**
  A list of the parameters and their values to be negotiated shall be provided in the Configuration Options field. These are defined in Section 5 on page 50. A Configuration Request 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.
4.5 CONFIGURATION RESPONSE (CODE 0X05)

Configuration Response packets shall be sent in reply to Configuration Request packets except when the error condition is covered by a Command Reject response. 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. For example, if a configuration request relates to traffic flowing from device A to device B, the sender of the configuration response may adjust this value for the same traffic flowing from device A to device B, but the response can not adjust the value in the reverse direction.

The options sent in the Response depend on the value in the Result field. Figure 4.8 on page 43 defines the format of the Configuration Response packet. See also Section 7.1 on page 76 for details of the configuration process.

Figure 4.8: Configuration Response Packet

The data fields are:

- **Source CID - SCID (2 octets)**
  This field contains the channel endpoint on the device receiving this Response packet. The device receiving the Response shall 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 4.9 displays the two-octet Flags field. Note the most significant bit is shown on the left.

Only one flag is defined, the Continuation flag (C).
More configuration responses will follow when C is set to one. 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.

The other flag bits are reserved and shall be set to zero. L2CAP implementations shall ignore these bits.

- **Result (2 octets)**
  
  The Result field indicates whether or not the Request was acceptable. See Table 4.6 on page 44 for possible result codes.

<table>
<thead>
<tr>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Success</td>
</tr>
<tr>
<td>0x0001</td>
<td>Failure – unacceptable parameters</td>
</tr>
<tr>
<td>0x0002</td>
<td>Failure – rejected (no reason provided)</td>
</tr>
<tr>
<td>0x0003</td>
<td>Failure – unknown options</td>
</tr>
<tr>
<td>Other</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

*Table 4.6: Configuration Response Result codes*

- **Configuration Options**
  
  This field contains the list of parameters being configured. These are defined in Section 5 on page 50. On a successful result, these parameters contain the return values for any wild card parameter values (see Section 5.3 on page 53) contained in the request.

  On an unacceptable parameters failure (Result=0x0001) the rejected parameters shall 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 accepted values and they too shall be included in the Configuration Response if they need to be changed.

  Each configuration parameter is one-directional. The configuration parameters describe the non default parameters the device sending the Configuration Request will accept. The configuration request can not request a change in the parameters the device receiving the request will accept.

  If a device needs to establish the value of a configuration parameter the remote device will accept, then it must wait for a configuration request containing that configuration parameter to be sent from the remote device.

  On an unknown option failure (Result=0x0003), the option types not understood by the recipient of the Request shall be included in the Response unless they are hints. Hints are those options in the Request that are skipped if not understood (see Section 5 on page 50). Hints shall not be included in the Response and shall 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 implementation specific.
4.6 DISCONNECTION REQUEST (CODE 0x06)

Terminating an L2CAP channel requires that a disconnection request be sent and acknowledged by a disconnection response. Figure 4.10 on page 45 shows a disconnection request. The receiver shall ensure that both source and destination CIDs match before initiating a channel disconnection.

Once a Disconnection Request is issued, all incoming data in transit on this L2CAP channel shall be discarded and any new additional outgoing data shall be discarded. Once a disconnection request for a channel has been received, all data queued to be sent out on that channel shall be discarded.

![Figure 4.10: Disconnection Request Packet](image)

The data fields are:

- **Destination CID - DCID (2 octets)**
  This field specifies the endpoint of the channel to be disconnected on the device receiving this request.

- **Source CID - SCID (2 octets)**
  This field specifies the endpoint of the channel to be disconnected on the device sending this request.

The SCID and DCID are relative to the sender of this request and shall 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 shall be sent in response. If the receiver finds a DCID match but the SCID fails to find the same match, the request should be silently discarded.
4.7 DISCONNECTION RESPONSE (CODE 0x07)

Disconnection responses shall be sent in response to each valid disconnection request.

![Disconnection Response Packet](image)

The data fields are:

- **Destination CID - DCID (2 octets)**
  This field identifies the channel endpoint on the device sending the response.

- **Source CID - SCID (2 octets)**
  This field identifies the channel endpoint on the device receiving the response.

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

4.8 ECHO REQUEST (CODE 0x08)

Echo requests are used to request a response from a remote L2CAP entity. These requests may be used for testing the link or for passing vendor specific information using the optional data field. L2CAP entities shall respond to a valid Echo Request packet with an Echo Response packet. The Data field is optional and implementation specific. L2CAP entities should ignore the contents of this field if present.

![Echo Request Packet](image)
4.9 ECHO RESPONSE (CODE 0x09)

An Echo response shall be sent upon receiving a valid Echo Request. The identifier in the response shall match the identifier sent in the Request. The optional and implementation specific data field may contain the contents of the data field in the Request, different data, or no data at all.

![Figure 4.13: Echo Response Packet](image)

4.10 INFORMATION REQUEST (CODE 0X0A)

Information requests are used to request implementation specific information from a remote L2CAP entity. L2CAP implementations shall respond to a valid Information Request with an Information Response. It is optional to send Information Requests.

An L2CAP implementation shall only use optional features or attribute ranges for which the remote L2CAP entity has indicated support through an Information Response. Until an Information Response which indicates support for optional features or ranges has been received only mandatory features and ranges shall be used.

![Figure 4.14: Information Request Packet](image)

The data fields are:

- **InfoType (2 octets)**

  The InfoType defines the type of implementation specific information being requested. See Section 4.11 on page 48 for details on the type of information requested.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>Connectionless MTU</td>
</tr>
<tr>
<td>0x0002</td>
<td>Extended features supported</td>
</tr>
<tr>
<td>Other</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

*Table 4.7: InfoType definitions*
4.11 INFORMATION RESPONSE (CODE 0X0B)

An information response shall be sent upon receiving a valid Information Request. The identifier in the response shall match the identifier sent in the Request. The data field shall contain the value associated with the InfoType field sent in the Request, or shall be empty if the InfoType is not supported.

<table>
<thead>
<tr>
<th>LSB</th>
<th>octet 0</th>
<th>octet 1</th>
<th>octet 2</th>
<th>octet 3</th>
<th>MSB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code=0x0B</td>
<td>Identifier</td>
<td>Length</td>
<td>InfoType</td>
<td>Result</td>
</tr>
<tr>
<td>Data (optional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.15: Information Response Packet

The data fields are:

- **InfoType (2 octets)**
  The InfoType defines the type of implementation specific information that was requested. This value shall be copied from the InfoType field in the Information Request.

- **Result (2 octets)**
  The Result contains information about the success of the request. If result is "Success", the data field contains the information as specified in Table 4.9 on page 49. If result is "Not supported", no data shall be returned.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Success</td>
</tr>
<tr>
<td>0x0001</td>
<td>Not supported</td>
</tr>
<tr>
<td>Other</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 4.8: Information Response Result values

- **Data (0 or more octets)**
  The contents of the Data field depends on the InfoType. For InfoType = 0x0001 the data field contains the remote entity’s 2-octet acceptable connectionless MTU. The default value is defined in Section 3.2 on page 24.

  For InfoType = 0x0002, the data field contains the 4 octet L2CAP extended feature mask. The feature mask refers to the extended features that the L2CAP entity sending the Information Response supports. The feature bits contained in the L2CAP feature mask are specified in Section 4.12 on page 49.
Note: L2CAP entities of versions prior to version 1.2, receiving an Information Request with InfoType = 0x0002 for an L2CAP feature discovery, return an Information Response with result code "Not supported". L2CAP entities at version 1.2 or later that have an all zero extended features mask may return an Information Response with result code "Not supported".

<table>
<thead>
<tr>
<th>InfoType</th>
<th>Data</th>
<th>Data Length (octets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>Connectionless MTU</td>
<td>2</td>
</tr>
<tr>
<td>0x0002</td>
<td>Extended feature mask</td>
<td>4</td>
</tr>
</tbody>
</table>

*Table 4.9: Information Response Data fields*

### 4.12 EXTENDED FEATURE MASK

The features are represented as a bit mask in the Information Response data field (see Section 4.11 on page 48). For each feature a single bit is specified which shall be set to 1 if the feature is supported and set to 0 otherwise. All unknown, reserved, or unassigned feature bits shall be set to 0.

The feature mask shown in Table 4.10 on page 49 consists of 4 octets (numbered octet 0 ... 3), with bit numbers 0 ... 7 each. Within the Information Response packet data field, bit 0 of octet 0 is aligned leftmost, bit 7 of octet 3 is aligned rightmost.

Note: the L2CAP feature mask is a new concept introduced in Bluetooth v1.2 and thus contains new features introduced after Bluetooth v1.1.

<table>
<thead>
<tr>
<th>No.</th>
<th>Supported feature</th>
<th>Octet</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Flow control mode</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Retransmission mode</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Bi-directional QoS¹</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Enhanced Retransmission Mode</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Streaming Mode</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>FCS Option</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>31</td>
<td>Reserved for feature mask extension</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

*Table 4.10: Extended feature mask.*

¹ Peer side supports upper layer control of the Link Manager's Bi-directional QoS, see Section 5.3 on page 53 for more details.
5 CONFIGURATION PARAMETER OPTIONS

Options are a mechanism to extend the configuration parameters. Options shall be transmitted as information elements containing an option type, an option length, and one or more option data fields. Figure 5.1 illustrates the format of an option.

![Figure 5.1: Configuration option format](image)

The configuration option fields are:

- **Type (1 octet)**
  - The option type field defines the parameters being configured. The most significant bit of the type determines the action taken if the option is not recognized.
  - 0 - option must be recognized; if the option is not recognized then refuse the configuration request
  - 1 - option is a hint; if the option is not recognized then skip the option and continue processing

- **Length (1 octet)**
  - The length field defines the number of octets in the option data. Thus an option type without option data has a length of 0.

- **Option data**
  - The contents of this field are dependent on the option type.

5.1 MAXIMUM TRANSMISSION UNIT (MTU)

This option specifies the maximum SDU size the sender of this option is capable of accepting for a channel. The type is 0x01, and the payload length is 2 octets, carrying the two-octet MTU size value as the only information element (see Figure 5.2 on page 51). Unlike the B-Frame length field, the I-Frame length field may be greater than the configured MTU because it includes the octet lengths of the Control, L2CAP SDU Length (when present), and frame check sequence fields as well as the Information octets.

MTU is not a negotiated value, it is an informational parameter that each device can specify independently. It indicates to the remote device that the local device can receive, in this channel, an MTU larger than the minimum required. All L2CAP implementations shall support a minimum MTU of 48 octets, however some protocols and profiles explicitly require support for a
larger MTU. The minimum MTU for a channel is the larger of the L2CAP minimum 48 octet MTU and any MTU explicitly required by the protocols and profiles using that channel. (Note: the MTU is only affected by the profile directly using the channel. For example, if a service discovery transaction is initiated by a non service discovery profile, that profile does not affect the MTU of the L2CAP channel used for service discovery).

The following rules shall be used when responding to a configuration request specifying the MTU for a channel:

- A request specifying any MTU greater than or equal to the minimum MTU for the channel shall be accepted.
- A request specifying an MTU smaller than the minimum MTU for the channel may be rejected.

The signaling described in Section 4.5 on page 43 may be used to reject an MTU smaller than the minimum MTU for a channel. The "failure-unacceptable parameters" result sent to reject the MTU shall include the proposed value of MTU that the remote device intends to transmit. It is implementation specific whether the local device continues the configuration process or disconnects the channel.

If the remote device sends a positive configuration response it shall include the actual MTU to be used on this channel for traffic flowing into the local device. This is the minimum of the MTU in the configuration request and the outgoing MTU capability of the device sending the configuration response. The new agreed value (the default value in a future re-configuration) is the value specified in the request.

The MTU to be used on this channel for the traffic flowing in the opposite direction will be established when the remote device sends its own Configuration Request as explained in Section 4.4 on page 40.

![Figure 5.2: MTU Option Format](image_url)
The option data field is:

- **Maximum Transmission Unit - MTU (2 octets)**
  
  The MTU field is the maximum SDU size, in octets, that the originator of the Request can accept for this 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 shall support a minimum MTU size of 48 octets. The default value is 672 octets.

### 5.2 FLUSH TIMEOUT OPTION

This option is used to inform the recipient of the Flush Timeout the sender is going to use. The Flush Timeout is defined in the Baseband specification “Flushing Payloads” on page 142[vol. 2]. The type is 0x02 and the payload size is 2 octets.

If the remote device returns a negative response to this option and the local device cannot honor the proposed value, then it shall either continue the configuration process by sending a new request with the original value, or disconnect the channel. The flush timeout applies to all channels on the same ACL logical transport but may be overridden on a packet by packet basis by marking individual L2CAP packets as non-automatically-flushable via the Packet_Boundary_Flag in the HCI ACL Data Packet (see section 1.1 on page 11).

![Figure 5.3: Flush Timeout option format.](image)

The option data field is:

- **Flush Timeout**

  This value is the Flush Timeout in milliseconds. 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.

  Possible values are:
  
  0x0001 - no retransmissions at the baseband level should be performed since the minimum polling interval is 1.25 ms.
  
  0x0002 to 0xFFFF - Flush Timeout used by the baseband.

---

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).
0xFFFF - an infinite amount of retransmissions. This is also referred to as a 'reliable channel'. In this case, the baseband shall continue retransmissions until physical link loss is declared by link manager timeouts.

5.3 QUALITY OF SERVICE (QOS) OPTION

This option specifies a flow specification similar to RFC 1363\(^1\). Although the RFC flow specification addresses only the transmit characteristics, the Bluetooth QoS interface can handle the two directions (Tx and Rx) in the negotiation as described below.

If no QoS configuration parameter is negotiated the link shall assume the default parameters. The QoS option is type 0x03.

In a configuration request, this option describes the outgoing traffic flow from the device sending the request. In a positive Configuration Response, this option describes the incoming traffic flow agreement to the device sending the response. In a negative Configuration Response, this option describes the preferred incoming traffic flow to 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 shall be assumed. If any QoS guarantees are required then a QoS configuration request shall be sent.

The remote device’s Configuration Response contains information that depends on the value of the result field (see Section 4.5 on page 43). If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is "Failure – unacceptable parameters", the response shall include a list of outgoing flow specification 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 accepted values.

If a configuration request contains any QoS option parameters set to "do not care" then the configuration response shall set the same parameters to "do not care". This rule applies for both Best Effort and Guaranteed Service.

The option data fields are:

- **Flags (1 octet)**
  Reserved for future use and shall be set to 0 and ignored by the receiver.

- **Service Type (1 octet)**
  This field indicates the level of service required. Table 5.1 on page 54 defines the different services available. The default value is 'Best effort'.

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

  If 'No traffic' is selected, the remainder of the fields shall be ignored because there is no data being sent across the channel in the outgoing direction.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>No traffic</td>
</tr>
<tr>
<td>0x01</td>
<td>Best effort (Default)</td>
</tr>
<tr>
<td>0x02</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Other</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

*Table 5.1: Service type definitions*

- **Token Rate (4 octets)**
  The value of this field represents the average data rate with which the application transmits data. The application may send data at this rate continuously. On a short time scale the application may send data in excess of the average data rate, dependent on the specified Token Bucket Size and Peak Bandwidth (see below). The Token Bucket Size and Peak Bandwidth allow the application to transmit data in a 'bursty' fashion.
The Token Rate signalled between two L2CAP peers is the data transmitted by the application and shall exclude the L2CAP protocol overhead. The Token Rate signalled over the interface between L2CAP and the Link Manager shall include the L2CAP protocol overhead. Furthermore the Token Rate value signalled over this interface may also include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

The Token Rate is the rate with which traffic credits are provided. Credits can be accumulated up to the Token Bucket Size. Traffic credits are consumed when data is transmitted by the application. When traffic is transmitted, and there are insufficient credits available, the traffic is non-conformant. The Quality of Service guarantees are only provided for conformant traffic. For non-conformant traffic there may not be sufficient resources such as bandwidth and buffer space. Furthermore non-conformant traffic may violate the QoS guarantees of other traffic flows.

The Token Rate is specified in octets per second. The value 0x00000000 indicates no token rate is specified. This is the default value and means “do not care”. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token rate available. The meaning of this value depends on 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 Token Bucket Size specifies a limit on the 'burstiness' with which the application may transmit data. The application may offer a burst of data equal to the Token Bucket Size instantaneously, limited by the Peak Bandwidth (see below). The Token Bucket Size is specified in octets.

The Token Bucket Size signalled between two L2CAP peers is the data transmitted by the application and shall exclude the L2CAP protocol overhead. The Token Bucket Size signalled over the interface between L2CAP and Link Manager shall include the L2CAP protocol overhead. Furthermore the Token Bucket Size value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

The value of 0x00000000 means that no token bucket is needed; this is the default value. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token bucket available. The meaning of this value depends on 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 L2CAP SDU size.

The Token Bucket Size is a property of the traffic carried over the L2CAP channel. The Maximum Transmission Unit (MTU) is a property of an L2CAP implementation. For the Guaranteed service the Token Bucket Size shall be smaller or equal to the MTU.
• **Peak Bandwidth (4 octets)**
  The value of this field, expressed in octets per second, limits how fast packets from applications may be sent back-to-back. Some systems can take advantage of this information, resulting in more efficient resource allocation.

  The Peak Bandwidth signalled between two L2CAP peers specifies the data transmitted by the application and shall exclude the L2CAP protocol overhead. The Peak Bandwidth signalled over the interface between L2CAP and Link Manager shall include the L2CAP protocol overhead. Furthermore the Peak Bandwidth value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The value of 0x00000000 means "don't care". This states that the device has no preference on incoming maximum bandwidth, and is the default value. When the Guaranteed service is selected, the default value shall not be used.

• **Access Latency (4 octets)**
  The value of this field is the maximum acceptable delay of an L2CAP packet to the air-interface. The precise interpretation of this number depends on over which interface this flow parameter is signalled. When signalled between two L2CAP peers, the Access Latency is the maximum acceptable delay between the instant when the L2CAP SDU is received from the upper layer and the start of the L2CAP SDU transmission over the air. When signalled over the interface between L2CAP and the Link Manager, it is the maximum delay between the instant the first fragment of an L2CAP PDU is stored in the Host Controller buffer and the initial transmission of the L2CAP packet on the air.

  Thus the Access Latency value may be different when signalled between L2CAP and the Link Manager to account for any queuing delay at the L2CAP transmit side. Furthermore the Access Latency value may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The Access Latency is expressed in microseconds. The value 0xFFFFFFFF means "do not care" and is the default value. When the Guaranteed service is selected, the default value shall not be used.

• **Delay Variation (4 octets)**
  The value of this field is the difference, in microseconds, between the maximum and minimum possible delay of an L2CAP SDU between two L2CAP peers. The Delay Variation is a purely informational parameter. The value 0xFFFFFFFF means "do not care" and is the default value.
5.4 RETRANSMISSION AND FLOW CONTROL OPTION

This option specifies whether retransmission and flow control is used. If the feature is used incoming parameters are specified by this option.

The option data fields are:

- **Mode (1 octet)**
  
The field contain the requested mode of the link. Possible values are shown in Table 5.2 on page 57.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Basic L2CAP mode</td>
</tr>
<tr>
<td>0x01</td>
<td>Retransmission mode</td>
</tr>
<tr>
<td>0x02</td>
<td>Flow control mode</td>
</tr>
<tr>
<td>0x03</td>
<td>Enhanced Retransmission mode</td>
</tr>
<tr>
<td>0x04</td>
<td>Streaming mode</td>
</tr>
<tr>
<td>Other values</td>
<td>Reserved for future use</td>
</tr>
</tbody>
</table>

Table 5.2: Mode definitions.

The Basic L2CAP mode is the default. If Basic L2CAP mode is requested then all other parameters shall be ignored.

Enhanced Retransmission mode shall be enabled if a reliable channel has been requested.

Enhanced Retransmission mode shall only be sent to an L2CAP entity that has previously advertised support for the mode in its Extended Feature Mask (see section 4.12).

Streaming mode should be enabled if a finite L2CAP Flush Time-Out is set on an L2CAP connection. Streaming mode shall only be sent to a device that has previously advertised support for the mode in the Extended Feature Mask (see section 4.12).

Basic mode, Flow Control mode and Retransmission mode shall only be used for backwards compatibility with L2CAP entities that do not support Enhanced Retransmission mode or Streaming mode.
• **TxWindow size (1 octet)**

This field specifies the size of the transmission window for Flow Control mode, Retransmission mode, and Enhanced Retransmission mode. The range is 1 to 32 for Flow Control mode and Retransmission mode. The range is 1 to 63 for Enhanced Retransmission mode.

In Retransmission mode and Flow Control mode this parameter should be negotiated to reflect the buffer sizes allocated for the connection on both sides. In general, the Tx Window size should be made as large as possible to maximize channel utilization. Tx Window size also controls the delay on flow control action. The transmitting device can send as many PDUs fit within the window.

In Enhanced Retransmission mode this value indicates the maximum number of I-frames that the sender of the option can receive without acknowledging some of the received frames. It is not negotiated. It is an informational parameter that each L2CAP entity can specify separately. In general, the TxWindow size should be made as large as possible to maximize channel utilization. The transmitting L2CAP entity can send as many PDUs as will fit within the receiving L2CAP entity's TxWindow. TxWindow size values in a Configuration Response indicate the maximum number of packets the sender can send before it requires an acknowledgement. In other words it represents the number of unacknowledged packets the send can hold. The value sent in a Configuration Response shall be less than or equal to the TxWindow size sent in the Configuration Request. The receiver of this option in the Configuration Response may use this value as part of its acknowledgement algorithm.

In Streaming mode this value is not used and shall be set to 0 and ignored by the receiving device.
• **MaxTransmit (1 octet)**

This field controls the number of transmissions of a single I-frame that L2CAP is allowed to try in Retransmission mode and Enhanced Retransmission mode. The minimum value is 1 (one transmission is permitted).

MaxTransmit controls the number of retransmissions that L2CAP is allowed to try in Retransmission mode and Enhanced Retransmission mode before accepting that a packet and the channel is lost. When a packet is lost after being transmitted MaxTransmit times the channel shall be disconnected by sending a Disconnect request (see section 4.6). In Enhanced Retransmission mode MaxTransmit controls the number of retransmissions for I-frames and S-frames with P-bit set to 1. The sender of the option in a Configuration Request specifies the value that shall be used by the receiver of the option. MaxTransmit values in a Configuration Response shall be ignored. Lower values might be appropriate for services requiring low latency. Higher values will be suitable for a link requiring robust operation. A value of 1 means that no retransmissions will be made but also means that the channel will be disconnected as soon as a packet is lost. MaxTransmit shall not be set to zero in Retransmission mode. In Enhanced Retransmission mode a value of zero for MaxTransmit means infinite retransmissions.

In Streaming mode this value is not used and shall be set to 0 and ignored by the receiving L2CAP entity.

• **Retransmission time-out (2 octets)**

This is the value in milliseconds of the retransmission time-out (this value is used to initialize the RetransmissionTimer).

The purpose of this timer in retransmission mode is to activate a retransmission in some exceptional cases. In such cases, any delay requirements on the channel may be broken, so the value of the timer should be set high enough to avoid unnecessary retransmissions due to delayed acknowledgments. Suitable values could be 100’s of milliseconds and up.

The purpose of this timer in flow control mode is to supervise I-frame transmissions. If an acknowledgement for an I-frame is not received within the time specified by the RetransmissionTimer value, either because the I-frame has been lost or the acknowledgement has been lost, the timeout will cause the transmitting side to continue transmissions. Suitable values are implementation dependent.

The purpose of this timer in Enhanced Retransmission mode is to detect lost I-frames and initiate appropriate error recovery. The value used for the Retransmission time-out is specified in Section 8.6.1. The value sent in a Configuration Request shall be set to 0. A value for the Retransmission time-out value shall be sent in a positive Configuration Response and indicates the value that will be used by the sender of the Configuration Response.

In Streaming mode this value is not used and shall be set to 0 and ignored by the receiving L2CAP entity.

• **Monitor time-out (2 octets)**
In Retransmission mode this is the value in milliseconds of the interval at which S-frames should be transmitted on the return channel when no frames are received on the forward channel. (this value is used to initialize the MonitorTimer, see below).

This timer ensures that lost acknowledgements are retransmitted. Its main use is to recover Retransmission Disable Bit changes in lost frames when no data is being sent. The timer shall be started immediately upon transitioning to the open state. It shall remain active as long as the connection is in the open state and the retransmission timer is not active. Upon expiration of the Monitor timer an S-frame shall be sent and the timer shall be restarted. If the monitor timer is already active when an S-frame is sent, the timer shall be restarted. An idle connection will have periodic monitor traffic sent in both directions. The value for this time-out should also be set to 100’s of milliseconds or higher.

In Enhanced Retransmission mode the Monitor time-out is used to detect lost S-frames with P-bit set to 1. If the time-out occurs before a response with the F-bit set to 1 is received the S-frame is resent. The value used for the Monitor time-out is specified in Section 8.6.3. The value sent in a Configuration Request shall be set to 0. A value for the Monitor time-out shall be sent in a positive Configuration Response and indicates the value that will be used by the sender of the Configuration Response.

In Streaming mode this value is not used and shall be set to 0 and ignored by the receiving device.

- Maximum PDU payload Size - MPS (2 octets)

The maximum size of payload data in octets that the L2CAP layer entity sending the option in a Configuration Request is capable of accepting, i.e. the MPS corresponds to the maximum PDU payload size. Each device specifies the value separately. An MPS value sent in positive Configuration Response is the actual MPS the receiver of the Configuration Request will use on this channel for traffic flowing into the local device. An MPS value sent in a positive Configuration Response shall be equal to or smaller than the value sent in the Configuration Request.

When using Retransmission mode and Flow Control mode the settings are configured separately for the two directions of an L2CAP connection. For example, in operating with an L2CAP entity implementing an earlier version of the core specification, an L2CAP connection can be configured as Flow Control mode in one direction and Retransmission mode in the other direction. If Basic L2CAP mode is configured in one direction and Retransmission mode or Flow control mode is configured in the other direction on the same L2CAP channel then the channel shall not be used.

Note: This asymmetric configuration only occurs during configuration.

When using Enhanced Retransmission mode or Streaming mode, both directions of the L2CAP connection shall be configured to the same mode. A prece-
The presence algorithm shall be used by both devices so a mode conflict can be resolved in a quick and deterministic manner.

There are two operating states:

- A device has a desired mode but is willing to use another mode (known as "state 1"), and
- A device requires a specific mode (known as "state 2")

In state 1, Basic L2CAP mode has the highest precedence and shall take precedence over Enhanced Retransmission mode and Streaming mode. Enhanced Retransmission mode has the second highest precedence and shall take precedence over all other modes except Basic L2CAP mode. Streaming mode shall have the next level of precedence after Enhanced Retransmission mode.

In state 2, a layer above L2CAP requires Enhanced Retransmission mode or Streaming mode. In this case, the required mode takes precedence over all other modes.

A device does not know in which state the remote device is operating so the state 1 precedence algorithm assumes that the remote device may be a state 2 device. If the mode proposed by the remote device has a higher precedence (according to the state 1 precedence) then the algorithm will operate such that creation of a channel using the remote device’s mode has higher priority than disconnecting the channel.

The algorithm for state 1 devices is divided into two parts. Part one covers the case where the remote device proposes a mode with a higher precedence than the state 1 local device. Part two covers the case where the remote device proposes a mode with a lower precedence than the state 1 local device. Part one of the algorithm is as follows:

- When the remote device receives the Configuration Request from the local device it will either reject the local device's Configuration Request by sending a negative Configuration Response or disconnect the channel. The negative Configuration Response will contain the remote device’s desired mode.
- Upon receipt of the negative Configuration Request the local device shall either send a second Configuration Response proposing the mode contained in the remote device’s negative Configuration Response or disconnect the channel.
- When the local device receives the Configuration Request from the remote device it shall send a positive Configuration Response or disconnect the channel.
- If the mode in the remote device’s negative Configuration Response does not match the mode in the remote device’s Configuration Request then the local device shall disconnect the channel.

Part two of the algorithm is as follows:
• When the local device receives the Configuration Request from the remote device it shall reject the Configuration Request by sending a negative Configuration Response proposing its desired mode. The local device's desired mode shall be the same mode it sent in its Configuration Request. Upon receiving the negative Configuration Response the remote device will either send a second Configuration Request or disconnect the channel.

• If the local device receives a second Configuration Request from the remote device that does not contain the desired mode then the local device shall disconnect the channel.

• If the local device receives a negative Configuration Response then it shall disconnect the channel.

An example of the algorithm for state 1 devices is as follows:

• The remote device proposes Basic L2CAP mode in a Configuration Request and the local device proposes Enhanced Retransmission mode or Streaming mode. The remote device rejects the local device's Configuration Request by sending a negative Configuration Response proposing Basic L2CAP mode. The local device will send a second Configuration Request proposing Basic L2CAP mode or disconnect the channel. If the local device sends a second Configuration Request that does not propose Basic L2CAP mode then the remote device will disconnect the channel. If the local device rejects the remote device's Configuration Request then the remote device will disconnect the channel.

The algorithm for state 2 devices is as follows:

• If the local device proposes a mode in a Configuration Request and the remote device proposes a different mode or rejects the local device's Configuration Request then the local device shall disconnect the channel.

For Enhanced Retransmission mode and Streaming mode the Retransmission and Flow Control option parameters shall not be changed in a subsequent reconfiguration after the channel has reached the OPEN state.

5.5 FRAME CHECK SEQUENCE (FCS) OPTION

The FCS option shall only be used when configuring Enhanced Retransmission mode and Streaming mode to specify the type of Frame Check Sequence (FCS) that will be included on S/I-Frames that are sent. The Frame Check Sequence option is type 0x05. "No FCS" shall only be used if both L2CAP entities send the FCS Option with value 0x00 (No FCS) in a configuration request. If one L2CAP entity sends the FCS Option with "No FCS" in a configuration request and the other L2CAP sends the FCS Option with a value other than "No FCS" then the default shall be used. If one or both L2CAP entities do not send the FCS option in a configuration request then the default shall be used.
The FCS types are shown in Table 5.3

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>No FCS</td>
</tr>
<tr>
<td>0x01</td>
<td>16-bit FCS defined in section 3.3.5 (default)</td>
</tr>
<tr>
<td>0x02 - 0xFF</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 5.3: FCS types

Value of 0x00 is set when the sender wishes to omit the FCS from S/I-Frames.
6 STATE MACHINE

This section is informative. The state machine may not represent all possible scenarios.

6.1 GENERAL RULES FOR THE STATE MACHINE:

• It is implementation specific, and outside the scope of this specification, how the transmissions are triggered.
• "Ignore" means that the signal can be silently discarded.

The following states have been defined to clarify the protocol; the actual number of states and naming in a given implementation is outside the scope of this specification:

• CLOSED – channel not connected.
• WAIT_CONNECT – a connection request has been received, but only a connection response with indication “pending” can be sent.
• WAIT_CONNECT_RSP – a connection request has been sent, pending a positive connect response.
• CONFIG – the different options are being negotiated for both sides; this state comprises a number of substates, see Section 6.1.3 on page 66
• OPEN – user data transfer state.
• WAIT_DISCONNECT – a disconnect request has been sent, pending a disconnect response.

Below the L2CAP_Data message corresponds to one of the PDU formats used on connection-oriented data channels as described in section 3, including PDUs containing B-frames, I-frames, S-frames.

Some state transitions and actions are triggered only by internal events effecting one of the L2CAP entity implementations, not by preceding L2CAP signaling messages. It is implementation-specific and out of the scope of this specification, how these internal events are realized; just for the clarity of specifying the state machine, the following abstract internal events are used in the state event tables, as far as needed:

• OpenChannel_Req – a local L2CAP entity is requested to set up a new connection-oriented channel.
• OpenChannel_Rsp – a local L2CAP entity is requested to finally accept or refuse a pending connection request.
• ConfigureChannel_Req – a local L2CAP entity is requested to initiate an outgoing configuration request.
• CloseChannel_Req – a local L2CAP entity is requested to close a channel.
• SendData_Req – a local L2CAP entity is requested to transmit an SDU.
• ReconfigureChannel_Req – a local L2CAP entity is requested to reconfigure the parameters of a connection-oriented channel.
There is a single state machine for each L2CAP connection-oriented channel that is active. A state machine is created for each new L2CAP_ConnectReq received. The state machine always starts in the CLOSED state.

To simplify the state event tables, the RTX and ERTX timers, as well as the handling of request retransmissions are described in Section 6.2 on page 72 and not included in the state tables.

L2CAP messages not bound to a specific data channel and thus not impacting a channel state (e.g. L2CAP_InformationReq, L2CAP_EchoReq) are not covered in this section.

The following states and transitions are illustrated in Figure 6.1 on page 74.

### 6.1.1 CLOSED state

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenChannel_req</td>
<td>-</td>
<td>Send L2CAP_ConnectReq</td>
<td>WAIT_CONNECT_RSP</td>
</tr>
<tr>
<td>L2CAP_ConnectReq</td>
<td>Normal, connection is possible</td>
<td>Send L2CAP_ConnectRsp (success)</td>
<td>CONFIG (substate WAIT_CONFIG)</td>
</tr>
<tr>
<td>L2CAP_ConnectReq</td>
<td>Need to indicate pending</td>
<td>Send L2CAP_ConnectRsp (pending)</td>
<td>WAIT_CONNECT</td>
</tr>
<tr>
<td>L2CAP_ConnectReq</td>
<td>No resource, not approved, etc.</td>
<td>Send L2CAP_ConnectRsp (refused)</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_ConfigReq</td>
<td>-</td>
<td>Send L2CAP_CommandReject (with reason Invalid CID)</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_ConfigRsp</td>
<td>-</td>
<td>Ignore</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_DisconnectReq</td>
<td>-</td>
<td>Send L2CAP_DisconnectRsp</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_DisconnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_Data</td>
<td>-</td>
<td>Ignore</td>
<td>CLOSED</td>
</tr>
</tbody>
</table>

*Table 6.1: CLOSED state event table.*

**Notes:**
- The L2CAP_ConnectReq message is not mentioned in any of the other states apart from the CLOSED state, as it triggers the establishment of a new channel, thus the branch into a new instance of the state machine.
### 6.1.2 WAIT_CONNECT_RSP state

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>Success indicated in result</td>
<td>Send L2CAP_ConfigReq</td>
<td>CONFIG (substate WAIT_CONFIG)</td>
</tr>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>Result pending</td>
<td>-</td>
<td>WAIT_CONNECT_RSP</td>
</tr>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>Remote side refuses connection</td>
<td>-</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_ConfigReq</td>
<td>-</td>
<td>Send L2CAP_CommandReject (with reason Invalid CID)</td>
<td>WAIT_CONNECT_RSP</td>
</tr>
<tr>
<td>L2CAP_ConfigRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT_RSP</td>
</tr>
<tr>
<td>L2CAP_DisconnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT_RSP</td>
</tr>
<tr>
<td>L2CAP_Data</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT_RSP</td>
</tr>
</tbody>
</table>

**Table 6.2: WAIT_CONNECT_RSP state event table.**

**Notes:**
- An L2CAP_DisconnectReq message is not included here, since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

### 6.1.3 WAIT_CONNECT state

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenChannel_Rsp</td>
<td>Pending connection request is finally acceptable</td>
<td>Send L2CAP_Connect_Rsp (success)</td>
<td>CONFIG (substate WAIT_CONFIG)</td>
</tr>
<tr>
<td>OpenChannel_Rsp</td>
<td>Pending connection request is finally refused</td>
<td>Send L2CAP_Connect_Rsp (refused)</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT</td>
</tr>
<tr>
<td>L2CAP_ConfigRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT</td>
</tr>
<tr>
<td>L2CAP_DisconnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT</td>
</tr>
<tr>
<td>L2CAP_Data</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONNECT</td>
</tr>
</tbody>
</table>

**Table 6.3: WAIT_CONNECT state event table.**

**Notes:**
- An L2CAP_DisconnectReq or L2CAP_ConfigReq message is not included here, since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.
6.1.4 CONFIG state

As it is also described in Section 7.1 on page 76, both L2CAP entities initiate a configuration request during the configuration process. This means that each device adopts an initiator role for the outgoing configuration request, and an acceptor role for the incoming configuration request. Configurations in both directions may occur sequentially, but can also occur in parallel.

The following substates are distinguished within the CONFIG state:

- **WAIT_CONFIG** – a device has sent or received a connection response, but has neither initiated a configuration request yet, nor received a configuration request with acceptable parameters.

- **WAIT_SEND_CONFIG** – for the initiator path, a configuration request has not yet been initiated, while for the response path, a request with acceptable options has been received.

- **WAIT_CONFIG_REQ_RSP** – for the initiator path, a request has been sent but a positive response has not yet been received, and for the acceptor path, a request with acceptable options has not yet been received.

- **WAIT_CONFIG_RSP** – the acceptor path is complete after having responded to acceptable options, but for the initiator path, a positive response on the recent request has not yet been received.

- **WAIT_CONFIG_REQ** – the initiator path is complete after having received a positive response, but for the acceptor path, a request with acceptable options has not yet been received.

According to Section 6.1.1 on page 65 and Section 6.1.2 on page 66, the CONFIG state is entered via WAIT_CONFIG substate from either the CLOSED state, the WAIT_CONNECT state, or the WAIT_CONNECT_RSP state. The CONFIG state is left for the OPEN state if both the initiator and acceptor paths complete successfully.

For better overview, separate tables are given: Table 6.4 shows the success transitions; therein, transitions on one of the minimum paths (no previous non-success transitions) are shaded. Table 6.5 on page 68 shows the non-success transitions within the configuration process, and Table 6.6 on page 69 shows further transition cause by events not belonging to the configuration process itself. The following configuration states and transitions are illustrated in Figure 6.2 on page 75.

<table>
<thead>
<tr>
<th>Previous state</th>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAIT_CONFIG</td>
<td>ConfigureChannel_Req</td>
<td>-</td>
<td>Send L2CAP_Config Req</td>
<td>WAIT_CONFIG_REQ_RSP</td>
</tr>
<tr>
<td>WAIT_CONFIG</td>
<td>L2CAP_ConfigReq</td>
<td>Options acceptable</td>
<td>Send L2CAP_Config Rsp (success)</td>
<td>WAIT_SEND_CONFIG</td>
</tr>
</tbody>
</table>

Table 6.4: CONFIG state/substates event table: success transitions within configuration process.
### Table 6.4: CONFIG state/substates event table: success transitions within configuration process.

<table>
<thead>
<tr>
<th>Previous state</th>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAIT_CONFIG</td>
<td>L2CAP_ConfigReq</td>
<td>Options acceptable</td>
<td>Send L2CAP_ConfigRsp (success)</td>
<td>WAIT_CONFIG_RSP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remote side accepts options</td>
<td>- (continue waiting for configuration request)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L2CAP_ConfigRsp</td>
<td>Options acceptable</td>
<td>Send L2CAP_ConfigRsp (success)</td>
<td>WAIT_CONFIG_REQ</td>
</tr>
<tr>
<td></td>
<td>ConfigureChannel_Req</td>
<td>-</td>
<td>Send L2CAP_ConfigRsp (success)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remote side accepts options</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.5: CONFIG state/substates event table: non-success transitions within configuration process.

<table>
<thead>
<tr>
<th>Previous state</th>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAIT_CONFIG</td>
<td>L2CAP_ConfigReq</td>
<td>Options not acceptable</td>
<td>Send L2CAP_ConfigRsp (fail)</td>
<td>WAIT_CONFIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_CONFIG</td>
</tr>
<tr>
<td></td>
<td>L2CAP_ConfigRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_SEND_CONFIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options not acceptable</td>
<td>Send L2CAP_ConfigRsp (fail)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remote side rejects options</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L2CAP_ConfigRsp</td>
<td>Options not acceptable</td>
<td>Send L2CAP_ConfigRsp (fail)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remote side rejects options</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Logical Link Control and Adaptation Protocol Specification**

**Bluetooth Core Specification Addendum 1 [vol 3]**
### Table 6.6: CONFIG state/substates event table: events not related to configuration process.

<table>
<thead>
<tr>
<th>Previous state</th>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG (any substate)</td>
<td>CloseChannel_Req</td>
<td>Any internal reason to stop</td>
<td>Send L2CAP_DISCONNECT_REQ</td>
<td>WAIT_DISCONNECT</td>
</tr>
<tr>
<td>CONFIG (any substate)</td>
<td>L2CAP Disconnect Req</td>
<td>-</td>
<td>Send L2CAP Disconnect Rsp</td>
<td>CLOSED</td>
</tr>
<tr>
<td>CONFIG (any substate)</td>
<td>L2CAP Disconnect Rsp</td>
<td>-</td>
<td>Ignore</td>
<td>CONFIG (remain in sub-state)</td>
</tr>
<tr>
<td>CONFIG (any substate)</td>
<td>L2CAP_Data</td>
<td>-</td>
<td>Process the PDU</td>
<td>CONFIG (remain in sub-state)</td>
</tr>
</tbody>
</table>

**Notes:**

- Receiving data PDUs (L2CAP_Data) in CONFIG state should be relevant only in case of a transition to a reconfiguration procedure (from OPEN state). Discarding the received data is allowed only in Retransmission Mode. Discarding an S-frame is allowed but not recommended. If a S-frame is discarded, the monitor timer will cause a new S-frame to be sent after a time out.
- Indicating a failure in a configuration response does not necessarily imply a failure of the overall configuration procedure; instead, based on the information received in the negative response, a modified configuration request may be triggered.
6.1.5 OPEN state

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendData_req</td>
<td>-</td>
<td>Send L2CAP_Data packet according to configured mode</td>
<td>OPEN</td>
</tr>
<tr>
<td>ReconfigureChannel_req</td>
<td>-</td>
<td>Complete outgoing SDU Send L2CAP_ConfigReq</td>
<td>CONFIG (sub-state WAIT_CONFIG_RSP)</td>
</tr>
<tr>
<td>CloseChannel_req</td>
<td>-</td>
<td>Send L2CAP_DisconnectReq</td>
<td>WAIT_DISCONNECT</td>
</tr>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>OPEN</td>
</tr>
<tr>
<td>L2CAP_ConfigReq</td>
<td></td>
<td>Complete outgoing SDU Send L2CAP_ConfigRsp</td>
<td>CONFIG (sub-state WAIT_CONFIG_REQ)</td>
</tr>
<tr>
<td>L2CAP_ConfigReq</td>
<td></td>
<td>Complete outgoing SDU Send L2CAP_ConfigRsp</td>
<td>OPEN</td>
</tr>
<tr>
<td>L2CAP_DisconnectReq</td>
<td>-</td>
<td>Send L2CAP_DisconnectRsp</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_DisconnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>OPEN</td>
</tr>
<tr>
<td>L2CAP_Data</td>
<td>-</td>
<td>Process the PDU</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

Table 6.7: OPEN state event table.

Note: The outgoing SDU shall be completed from the view of the remote entity. Therefore all PDUs forming the SDU shall have been reliably transmitted by the local entity and acknowledged by the remote entity, before entering the configuration state.

6.1.6 WAIT_DISCONNECT state

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2CAP_ConnectRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_DISCONNECT</td>
</tr>
<tr>
<td>L2CAP_ConfigReq</td>
<td>-</td>
<td>Send L2CAP_CommandReject with reason Invalid CID</td>
<td>WAIT_DISCONNECT</td>
</tr>
<tr>
<td>L2CAP_ConfigRsp</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_DISCONNECT</td>
</tr>
<tr>
<td>L2CAP_DisconnectReq</td>
<td>-</td>
<td>Send L2CAP_DisconnectRsp</td>
<td>CLOSED</td>
</tr>
<tr>
<td>L2CAP_DisconnectRsp</td>
<td>-</td>
<td>-</td>
<td>CLOSED</td>
</tr>
</tbody>
</table>

Table 6.8: WAIT_DISCONNECT state event table.
Table 6.8: WAIT_DISCONNECT state event table.

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2CAP_Data</td>
<td>-</td>
<td>Ignore</td>
<td>WAIT_DISCONNECT</td>
</tr>
</tbody>
</table>
6.2 TIMERS EVENTS

6.2.1 RTX

The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signaling requests. This timer is started when a signaling request (see Section 7 on page 76) 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 shall be reset to a new value at least double the previous value. When retransmitting the Request message, the context of the same state shall be assumed as with the original transmission. If a Request message is received that is identified as a duplicate (retransmission), it shall be processed in the context of the same state which applied when the original Request message was received.

Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before terminating the channel identified by the Requests. The one exception is the signaling CID that should never be terminated. The decision should be based on the flush timeout of the signaling 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. When terminating the channel, it is not necessary to send a L2CAP_DISCONNECTReq and enter WAIT_DISCONNECT state. Channels can be transitioned directly to the CLOSED state.

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 shall exist for each outstanding signaling 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 termination (if no response is received) is 60 seconds.
6.2.2 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_ConnectRsp event with a "connect pending" result (0x0001) 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 disconnected.

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 termination (if no response is received) is 300 seconds. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq and enter WAIT_DISCONNECT state. Channels should be transitioned directly to the CLOSED state.
Figure 6.1: States and transitions.
Figure 6.2: Configuration states and transitions.
7 GENERAL PROCEDURES

This section describes the general operation of L2CAP, including the configuration process, the handling and the processing of user data for transportation over the air interface. This section also describes the operation of L2CAP features including the delivery of erroneous packets, the flushing of expired data and operation in connectionless mode.

Procedures for the flow control and retransmission modes are described in Section 8 on page 84.

7.1 CONFIGURATION PROCESS

Configuring the channel parameters shall be done independently for both directions. Both configurations may be done in parallel. For each direction the following procedure shall be used:

1. Informing the remote side of the non-default parameters that the local side will accept using a Configuration Request

2. Remote side responds, agreeing or disagreeing with these values, including the default ones, using a Configuration Response.

3. The local and remote devices repeat steps (1) and (2) until agreement on all parameters is reached.

This process can be abstracted into the initial Request configuration path and a Response configuration path, followed by the reverse direction phase. Reconfiguration follows a similar two-phase process by requiring configuration in both directions.

The decision on the amount of time (or messages) spent configuring the channel parameters before terminating the configuration is left to the implementation, but it shall not last more than 120 seconds.
7.1.1 Request path

The Request Path can configure the following:
- requester’s incoming MTU.
- requester’s outgoing flush timeout.
- requester’s outgoing QoS.
- requester’s incoming flow and error control information.

Table 7.1 on page 77 defines the configuration options that may be placed in a Configuration Request.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTU</td>
<td>Incoming MTU information</td>
</tr>
<tr>
<td>FlushTO</td>
<td>Outgoing flush timeout</td>
</tr>
<tr>
<td>QoS</td>
<td>Outgoing QoS information</td>
</tr>
<tr>
<td>RFCMode</td>
<td>Incoming Retransmission and Flow Control Mode</td>
</tr>
</tbody>
</table>

Table 7.1: Parameters allowed in Request

The state machine for the configuration process is described in Section 6 on page 64.

7.1.2 Response path

The Response Path can configure the following:
- responder’s outgoing MTU, that is the remote side’s incoming MTU.
- remote side’s flush timeout.
- responder’s Outgoing Flow and Error Control information

If a request-oriented parameter is not present in the Request message (reverts to last agreed value), the remote side may negotiate for a non-default value by including the proposed value in a negative Response message.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTU</td>
<td>Outgoing MTU information</td>
</tr>
<tr>
<td>FlushTO</td>
<td>Incoming flush timeout</td>
</tr>
<tr>
<td>QoS</td>
<td>Incoming QoS information</td>
</tr>
<tr>
<td>RFCMode</td>
<td>Outgoing Retransmission and Flow Control Mode</td>
</tr>
</tbody>
</table>

Table 7.2: Parameters allowed in Response
7.2   FRAGMENTATION AND RECOMBINATION

Fragmentation is the breaking down of PDUs into smaller pieces for delivery from L2CAP to the lower layer. Recombination is the process of reassembling a PDU from fragments delivered up from the lower layer. Fragmentation and Recombination may be applied to any L2CAP PDUs.

7.2.1   Fragmentation of L2CAP PDUs

An L2CAP implementation may fragment any L2CAP PDU for delivery to the lower layers. If L2CAP runs directly over the link controller protocol, then an implementation may fragment the PDU into multiple baseband packets for transmission over the air. If L2CAP runs above the host controller interface, then an implementation may send HCI transport sized fragments to the Controller which passes them to the baseband. All L2CAP fragments associated with an L2CAP PDU shall be passed to the baseband before any other L2CAP PDU for the same logical transport shall be sent.

The two LLID bits defined in the first octet of baseband payload (also called the frame header) are used to signal the start and continuation of L2CAP PDUs. LLID shall be ‘10’ for the first segment in an L2CAP PDU and ‘01’ for a continuation segment. An illustration of fragmentation is shown in Figure 7.1 on page 78. An example of how fragmentation might be used in a device with HCI is shown in Figure 7.2 on page 79.

![Figure 7.1: L2CAP fragmentation.](image)

Note: The link controller is able to impose a different fragmentation on the PDU by using "start" and "continuation" indications as fragments are translated into baseband packets. Thus, both L2CAP and the link controller use the same mechanism to control the size of fragments.
7.2.2 Recombination of L2CAP PDUs

The link controller protocol attempts to deliver ACL packets in sequence and protects the integrity of the data using a 16-bit CRC. When errors are detected by the baseband it uses an automatic repeat request (ARQ) mechanism.

Recombination of fragments may occur in the Controller but ultimately it is the responsibility of L2CAP to reassemble PDUs and SDUs and to check the length field of the SDUs. As the baseband controller receives ACL packets, it either signals the L2CAP layer on the arrival of each baseband packet, or accumulates a number of packets (before the receive buffer fills up or a timer expires) before passing fragments to the L2CAP layer.

An L2CAP implementation shall use the length field in the header of L2CAP PDUs, see Section 3 on page 23, as a consistency check and shall discard any L2CAP PDUs that fail to match the length field. If channel reliability is not needed, packets with invalid lengths may be silently discarded. For reliable channels, an L2CAP implementation shall 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 5.2 on page 52. For higher data integrity L2CAP should be operated in the Retransmission Mode.

---

**Figure 7.2: Example of fragmentation processes in a device with HCI.**
7.3 ENCAPSULATION OF SDUs

All SDUs are encapsulated into one or more L2CAP PDUs.

In Basic L2CAP mode, an SDU shall be encapsulated with a minimum of L2CAP protocol elements, resulting in a type of L2CAP PDU called a Basic Information Frame (B-frame).

Segmentation and Reassembly operations are only used in Enhanced Retransmission mode, Streaming mode, Retransmission mode, and Flow Control mode. SDUs may be segmented into a number of smaller packets called SDU segments. Each segment shall be encapsulated with L2CAP protocol elements resulting in an L2CAP PDU called an Information Frame (I-frame).

The maximum size of an SDU segment shall be given by the Maximum PDU Payload Size (MPS). The MPS parameter may be exported using an implementation specific interface to the upper layer.

Note that this specification does not have a normative service interface with the upper layer, nor does it assume any specific buffer management scheme of a host implementation. Consequently, a reassembly buffer may be part of the upper layer entity. It is assumed that SDU boundaries shall be preserved between peer upper layer entities.

7.3.1 Segmentation of L2CAP SDUs

In Flow Control, Streaming, or Retransmission modes, incoming SDUs may be broken down into segments, which shall then be individually encapsulated with L2CAP protocol elements (header and checksum fields) to form I-frames. I-frames are subject to flow control and may be subject to retransmission procedures. The header carries a 2 bit SAR field that is used to identify whether the I-frame is a 'start', 'end' or 'continuation' packet or whether it carries a complete, unsegmented SDU. Figure 7.3 on page 80 illustrates segmentation and fragmentation.

![Figure 7.3: Segmentation and fragmentation of an SDU.](image-url)
### 7.3.2 Reassembly of L2CAP SDUs

The receiving side uses the SAR field of incoming 'I-frames' for the reassembly process. The L2CAP SDU length field, present in the "start of SDU" I-frame, is an extra integrity check, and together with the sequence numbers may be used to indicate lost L2CAP SDUs to the application. Figure 7.3 on page 80 illustrates segmentation and fragmentation.

### 7.3.3 Segmentation and fragmentation

Figure 7.4 on page 81 illustrates the use of segmentation and fragmentation operations to transmit a single SDU. Note that while SDUs and L2CAP PDUs are transported in peer-to-peer fashion, the fragment size used by the Fragmentation and Recombination routines is implementation specific and may not be the same in the sender and the receiver. The over-the-air sequence of baseband packets as created by the sender is common to both devices.

---

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.
7.4 DELIVERY OF ERRONEOUS L2CAP SDUS

Some applications may require corrupted or incomplete L2CAP SDUs to be delivered to the upper layer. If delivery of erroneous L2CAP SDUs is enabled, the receiving side will pass information to the upper layer on which parts of the L2CAP SDU (i.e., which L2CAP frames) have been lost, failed the error check, or passed the error check. If delivery of erroneous L2CAP SDUs is disabled, the receiver shall discard any L2CAP SDU segment with any missing frames or any frames failing the error checks. L2CAP SDUs whose length field does not match the actual frame length shall also be discarded.

7.5 OPERATION WITH FLUSHING

In the L2CAP configuration, the Flush Time-Out may be set separately per L2CAP channel, but in the baseband, the flush mechanisms operate per ACL logical transport.

When there is more than one L2CAP channel mapped to the same ACL logical transport, the automatic flush time-out does not discriminate between L2CAP channels. The exception is packets marked as non-automatically-flushable via the Packet_Boundary_Flag in the HCI ACL Data Packet (see section 1.1 on page 11). The automatic flush time-out flushes a specific automatically-flushable L2CAP PDU. The HCI Flush command flushes all outstanding L2CAP PDUs for the ACL logical transport including L2CAP PDUs marked as non-automatically-flushable. Therefore, care has to be taken when using the Automatic Flush Time-out and the HCI Flush command. The HCI Enhanced Flush command should be used instead:

1. For any connection to be reliable at the L2CAP level, it shall either mark all the packets associated with the reliable connection as non-automatically-flushable (if it is mapped to an ACL logical transport with a finite automatic flush time-out) or use L2CAP Enhanced Retransmission or Retransmission mode. In Enhanced Retransmission or Retransmission mode, loss of flushed L2CAP PDUs on the channel is detected by the L2CAP ARQ mechanism and they are retransmitted. Another approach is shown in 4 below. Note that L2CAP Enhanced Retransmission or Retransmission mode might be used for other purposes such as the need for a residual error rate that is much smaller than the baseband can deliver. In this situation L2CAP Enhanced Retransmission or Retransmission mode and the Non-Flushable Packet Boundary Flag feature can be used at the same time.

2. There is only one automatic flush time-out setting per ACL logical transport. Therefore, all time bounded L2CAP channels on an ACL logical transport with an automatic flush time-out setting should configure the same flush time-out value at the L2CAP level.

3. If Automatic Flush Time-out is used, then it should be taken into account that it only flushes one L2CAP PDU. If one PDU has timed out and needs flushing, then other automatically-flushable packets on the same logical transport are also likely to need flushing. Therefore, flushing can be handled by the
HCI Enhanced Flush command so that all outstanding automatically-flushable L2CAP PDUs are flushed.

4. When both reliable and isochronous data is to be sent over the same ACL logical transport, an infinite Automatic Flush Time-out can be used. In this case the isochronous data is flushed using the HCI Enhanced Flush command with Packet_Type set to "Automatically-Flushable Only," thus preserving the reliable data.

7.6 CONNECTIONLESS DATA CHANNEL

In addition to connection-oriented channels, L2CAP also has a connectionless channel. The connectionless channel allows transmission to all members of the piconet. Data sent through the connectionless channel is sent in a best-effort manner. The connectionless channel has no quality of service and is unreliable. L2CAP makes no guarantee that data sent through the connectionless channel successfully reaches all members of the piconet. If reliable group transmission is required, it must be implemented at a higher layer.

Transmissions to the connectionless channel will be sent to all members of the piconet. If this data is not for transmission to all members of the piconet, then higher level encryption is required to support private communication.

The local device will not receive transmissions on the connectionless channel, therefore, higher layer protocols must loopback any data traffic being sent to the local device.

An L2CAP service interface could provide basic group management mechanisms including creating a group, adding members to a group, and removing members from a group.

Connectionless data channels shall not be used with Enhanced Retransmission or Retransmission Mode or Flow Control Mode.
8 PROCEDURES FOR FLOW CONTROL AND RETRANSMISSION

When Enhanced Retransmission mode, Streaming mode, Flow Control mode, or Retransmission mode is used, the procedures defined in this chapter shall be used, including the numbering of information frames, the handling of SDU segmentation and reassembly, and the detection and notification of errored frames. Retransmission mode and Enhanced Retransmission mode also allow the sender to resend errored frames on request from the receiver.

8.1 INFORMATION RETRIEVAL

Before attempting to configure Enhanced Retransmission mode, Streaming mode, Flow Control mode, or Retransmission mode on a channel, support for the suggested mode shall be verified by performing an information retrieval for the "Extended features supported" information type (0x0002). If the information retrieval is not successful or the "Extended features mask" bit is not set for the wanted mode, the mode shall not be suggested in a configuration request.

8.2 FUNCTION OF PDU TYPES FOR FLOW CONTROL AND RETRANSMISSION

Two frame formats are defined for Enhanced Retransmission mode, Streaming Mode, Flow Control Mode, and Retransmission mode (see Section 3.3 on page 25). The I-frame is used to transport user information instead of the B-frame. The S-frame is used for supervision.

8.2.1 Information frame (I-frame)

I-frames are sequentially numbered frames containing information fields. I-frames also include some of the functionality of RR frames (see below).

8.2.2 Supervisory Frame (S-frame)

The S-frame is used to control the transmission of I-frames. For Retransmission mode and Flow Control mode, the S-frame has two formats: Receiver Ready (RR) and Reject (REJ). A description of how S-frames are used in Enhanced Retransmission mode is given in Section 8.6.1. S-frames are not used in Streaming mode. The following description of S-frames only applies to Retransmission mode and Flow Control mode.

8.2.2.1 Receiver Ready (RR)

The receiver ready (RR) S-frame is used to:

1. Acknowledge I-frames numbered up to and including ReqSeq - 1.
2. Enable or disable retransmission of I-frames by updating the receiver with the current status of the Retransmission Disable Bit.
The RR frame has no information field.

8.2.2.2 Reject (REJ)

The reject (REJ) S-frame is used to request retransmission of all I-frames starting with the I-frame with TxSeq equal to ReqSeq specified in the REJ. The value of ReqSeq in the REJ frame acknowledges I-frames numbered up to and including ReqSeq - 1. I-frames that have not been transmitted, shall be transmitted following the retransmitted I-frames.

When a REJ is transmitted, it triggers a REJ Exception condition. A second REJ frame shall not be transmitted until the REJ Exception condition is cleared. The receipt of an I-frame with a TxSeq equal to the ReqSeq of the REJ frame clears the REJ Exception. The REJ Exception condition only applies to traffic in one direction. Note: this means that only valid I-frames can be rejected.
8.3 VARIABLES AND SEQUENCE NUMBERS

The sending peer uses the following variables and Sequence numbers:

• TxSeq – the send Sequence number used to sequentially number each new I-frame transmitted.

• NextTxSeq – the Sequence number to be used in the next new I-frame transmitted.

• ExpectedAckSeq – the Sequence number of the next I-frame expected to be acknowledged by the receiving peer.

The receiving peer uses the following variables and Sequence numbers:

• ReqSeq – The Sequence number sent in an acknowledgement frame to request transmission of I-frame with TxSeq = ReqSeq and acknowledge receipt of I-frames up to and including (ReqSeq-1)

• ExpectedTxSeq – the value of TxSeq expected in the next I-frame.

• BufferSeq – When segmented I-frames are buffered this is used to delay acknowledgement of received I-frame so that new I-frame transmissions do not cause buffer overflow.

All variables have the range 0 to 63. Arithmetic operations on state variables (NextTXSeq, ExpectedTxSeq, ExpectedAckSeq, BufferSeq) and sequence numbers (TxSeq, ReqSeq) contained in this document shall be taken modulo 64.

To perform Modulo 64 operation on negative numbers multiples of 64 shall be added to the negative number until the result becomes non-negative.

8.3.1 Sending peer

8.3.1.1 Send sequence number TxSeq

I-frames contain TxSeq, the send sequence number of the I-frame. When an I-frame is first transmitted, TxSeq is set to the value of the send state variable NextTXSeq. TxSeq is not changed if the I-frame is retransmitted.

8.3.1.2 Send state variable NextTXSeq

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. A send state variable NextTxSeq shall be maintained for each remote endpoint. NextTxSeq is the sequence number of the next in-sequence I-frame to be transmitted to that remote endpoint. When the link is created NextTXSeq shall be initialized to 0.

The value of NextTxSeq shall be incremented by 1 after each in-sequence I-frame transmission, and shall not exceed ExpectedAckSeq by more than the maximum number of outstanding I-frames (TxWindow). The value of TxWindow shall be in the range 1 to 32 for Retransmission mode and Flow Control
mode. The value of TxWindow shall be in the range of 1 to 63 for Enhanced Retransmission mode.
8.3.1.3 Acknowledge state variable ExpectedAckSeq

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. An acknowledge state variable ExpectedAckSeq shall be maintained for each remote endpoint. ExpectedAckSeq is the sequence number of the next in-sequence I-frame that the remote receiving peer is expected to acknowledge. (ExpectedAckSeq – 1 equals the TxSeq of the last acknowledged I-frame). When the link is created ExpectedAckSeq shall be initialized to 0.

Note that if the next acknowledgement acknowledges a single I-frame then its ReqSeq will be expectedAckSeq + 1.

If a valid ReqSeq is received from the peer then ExpectedAckSeq is set to ReqSeq. A valid ReqSeq value is one that is in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

Note: The comparison with NextTXSeq must be ≤ in order to handle the situations where there are no outstanding I-frames.

These inequalities shall be interpreted in the following way: ReqSeq is valid, if and only if (ReqSeq-ExpectedAckSeq) mod 64 ≤ (NextTXSeq-ExpectedAckSeq) mod 64. Furthermore, from the description of NextTXSeq, it can be seen that (NextTXSeq-ExpectedAckSeq) mod 64 ≤ TxWindow.

Figure 8.1: Example of the transmitter side

Figure 8.1 on page 88 shows TxWindow=5, and three frames awaiting transmission. The frame with number F7 may be transmitted when the frame with F2 is acknowledged. When the frame with F7 is transmitted, TxSeq is set to the value of NextTXSeq. After TxSeq has been set, NextTxSeq is incremented.

The sending peer expects to receive legal ReqSeq values, these are in the range ExpectedAckSeq up to and including NextTxSeq. Upon receipt of a ReqSeq value equal to the current NextTxSeq all outstanding I-frames have been acknowledged by the receiver.
8.3.2 Receiving peer

8.3.2.1 Receive sequence number ReqSeq

All I-frames and S-frames contain ReqSeq, the send Sequence number (TxSeq) that the receiving peer requests in the next I-frame.

When an I-frame or an S-frame is transmitted, the value of ReqSeq shall be set to the current value of the receive state variable ExpectedTxSeq or the buffer state variable BufferSeq. The value of ReqSeq shall indicate that the data link layer entity transmitting the ReqSeq has correctly received all I-frames numbered up to and including ReqSeq – 1.

Note: The option to set ReqSeq to BufferSeq instead of ExpectedTxSeq allows the receiver to impose flow control for buffer management or other purposes. In this situation, if BufferSeq<>ExpectedTxSeq, the receiver should also set the retransmission disable bit to 1 to prevent unnecessary retransmissions.

8.3.2.2 Receive state variable, ExpectedTxSeq

Each channel shall have a receive state variable (ExpectedTxSeq). The receive state variable is the sequence number (TxSeq) of the next in-sequence I-frame expected.

The value of the receive state variable shall be the last in-sequence, valid I-frame received.

8.3.2.3 Buffer state variable BufferSeq

Each channel may have an associated BufferSeq. BufferSeq is used to delay acknowledgement of frames until they have been pulled by the upper layers, thus preventing buffer overflow. BufferSeq and ExpectedTxSeq are equal when there is no extra segmentation performed and frames are pushed to the upper layer immediately on reception. When buffer space is scarce, for example when frames reside in the buffer for a period, the receiver may choose to set ReqSeq to BufferSeq instead of ExpectedTxSeq, incrementing BufferSeq as buffer space is released. The windowing mechanism will ensure that transmission is halted when ExpectedTxSeq - BufferSeq is equal to TxWindow.

Note: Owing to the variable size of I-frames, updates of BufferSeq may be based on changes in available buffer space instead of delivery of I-frame contents.

I-Frames shall have sequence numbers in the range $\text{ExpectedTxSeq} \leq \text{TxSeq} < (\text{BufferSeq} + \text{TxWindow})$. 
On receipt of an I-frame with TxSeq equal to ExpectedTxSeq, ExpectedTxSeq shall be incremented by 1 regardless of how many I-frames with TxSeq greater than ExpectedTxSeq were previously received.

**Figure 8.2: Example of the receiver side**

Figure 8.2 on page 90 shows TxWindow=5. F1 is successfully received and pulled by the upper layer. BufferSeq shows that F2 is the next I-frame to be pulled, and ExpectedTxSeq points to the next I-frame expected from the peer. An I-frame with TxSeq equal to 5 has been received thus triggering an SREJ or REJ exception. The star indicates I-frames received but discarded owing to the SREJ or REJ exception. They will be resent as part of the error recovery procedure.

In Figure 8.2 there are several I-frames in a buffer awaiting the SDU reassembly function to pull them and the TxWindow is full. The receiver would usually disable retransmission in Retransmission mode or Flow Control mode by setting the Retransmission Disable Bit to 1 and send an RR back to the sending side. This tells the transmitting peer that there is no point in performing retransmissions. Both sides will send S-frames to make sure the peer entity knows the current value of the Retransmission Disable Bit.
8.4 RETRANSMISSION MODE

8.4.1 Transmitting frames

A new I-frame shall only be transmitted when the TxWindow is not full. No I-frames shall be transmitted if the last RetransmissionDisableBit (R) received is set to one.

A previously transmitted I-frame may be retransmitted as a result of an error recovery procedure, even if the TxWindow is full. When an I-frame is retransmitted it shall always be sent with the same TxSeq value used in its initial transmission.

The state of the RetransmissionDisableBit (R) is stored and used along with the state of the RetransmissionTimer to decide the actions when transmitting I-frames. The RetransmissionTimer is running whenever I-frames have been sent but not acknowledged.

8.4.1.1 Last received R was set to zero

If the last R received was set to zero, then I-frames may be transmitted. If there are any I-frames which have been sent and not acknowledged then they shall be retransmitted when the RetransmissionTimer elapses. If the retransmission timer has not elapsed then a retransmission shall not be sent and only new I-frames may be sent.

a) If unacknowledged I-frames have been sent and the RetransmissionTimer has elapsed then an unacknowledged I-frame shall be retransmitted. The RetransmissionTimer shall be restarted.

b) If unacknowledged I-frames have been sent, the Retransmission timer has not elapsed then a new I-frame shall be sent if one is waiting and no timer action shall be taken.

c) If no unacknowledged I-frames have been sent, and a new I-frame is waiting, then the new I-frame shall be sent, the RetransmissionTimer shall be started and if the MonitorTimer is running, it shall be stopped.

d) If no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted, and the RetransmissionTimer is running, then the retransmission timer shall be stopped and the monitor timer shall be started.
The table below summarizes actions when the RetransmissionTimer is in use and R=0.

| Table 8.1: Summary of actions when the RetransmissionTimer is in use and R=0. |
|---------------------------------|----------------|-----------------|-----------------|-----------------|
| Unacknowledged                 | Retransmission Timer         | New I-frames are waiting | Transmit Action  | Timer Action    |
| I-frames sent                   | has elapsed                 |                             |                 |                 |
| True                            | True                        | True or False              | Retransmit unacknowledged I-frame | Restart Retransmission Timer |
| True                            | False                       | True                        | Transmit new I-frame | No timer action |
| True                            | False                       | False                      | No transmit action | No Timer action |
| False                           | False                       | True                        | Transmit new I-frame | Restart Retransmission Timer |
| False                           | False                       | False                      | No Transmit action | If MonitorTimer is not running then restart MonitorTimer |

If the RetransmissionTimer is not in use, no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted
   a) If the MonitorTimer is running and has not elapsed then no transmit action shall be taken and no timer action shall be taken.
   b) If the MonitorTimer has elapsed then an S-frame shall be sent and the MonitorTimer shall be restarted.

If any I-frames become available for transmission then the MonitorTimer shall be stopped, the RetransmissionTimer shall be started and the rules for when the RetransmissionTimer is in use shall be applied.

When an I-frame is sent ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTxSeq and NextTxSeq shall be incremented by one.
8.4.1.2 Last received R was set to one

If the last R received was set to one, then I-frames shall not be transmitted. The only frames which may be sent are S-frames. An S-frame shall be sent according to the rules below:

a) If the MonitorTimer is running and has not elapsed then no transmit action shall be taken and no timer action shall be taken.

b) If the MonitorTimer has elapsed then an S-frame shall be sent and the MonitorTimer shall be restarted.

8.4.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be accepted for the SDU reassembly function. ExpectedTxSeq is used by the reassembly function.

The first valid I-frame received after an REJ was sent, with a TxSeq of the received I-frame equal to ReqSeq of the REJ, shall clear the REJ Exception condition.

The ReqSeq shall be processed according to Section 8.4.6 on page 95.

If a valid I-frame with TxSeq ≠ ExpectedTxSeq is received then an exception condition shall be triggered which is handled according to Section 8.4.7 on page 95.

8.4.3 I-frames pulled by the SDU reassembly function

When the L2CAP layer has removed one or more I-frames from the buffer, BufferSeq may be incremented in accordance with the amount of buffer space released. If BufferSeq is incremented, an acknowledgement shall be sent to the peer entity.

Note: Since the primary purpose of BufferSeq is to prevent buffer overflow, an implementation may choose to set BufferSeq in accordance with how many new incoming I-frames could be stored rather than how many have been removed.

The acknowledgement may either be an RR or an I-frame. The acknowledgement shall be sent to the peer L2CAP entity with ReqSeq equal to BufferSeq. When there are no I-frames buffered for pulling ExpectedTxSeq is equal to BufferSeq.

If the MonitorTimer is active then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.
8.4.4 Sending and receiving acknowledgements

Either the MonitorTimer or the RetransmissionTimer shall be active while in Retransmission Mode. Both timers shall not be active concurrently.

8.4.4.1 Sending acknowledgements

Whenever an L2CAP entity transmits an I-frame or an S-frame, ReqSeq shall be set to ExpectedTxSeq or BufferSeq.

8.4.4.2 Receiving acknowledgements

On receipt of a valid S-frame or I-frame, the ReqSeq contained in the frame shall acknowledge previously transmitted I-frames. ReqSeq acknowledges I-frames with a TxSeq up to and including ReqSeq – 1.

The following rules shall be applied:

1. If the RetransmissionDisableBit changed value from 0 to 1 (stop retransmissions) then the receiving entity shall
   a) If the RetransmissionTimer is running then stop it and start the MonitorTimer.
   b) Store the state of the RetransmissionDisableBit received.

2. If the RetransmissionDisableBit changed value from 1 to 0 (start retransmissions) then the receiving entity shall
   a) Store the state of the RetransmissionDisableBit received.
   b) If there are any I-frames that have been sent but not acknowledged, then stop the MonitorTimer and start the RetransmissionTimer.
   c) Any buffered I-frames shall be transmitted according to Section 8.4.1 on page 91.

3. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame, and the RetransmissionDisableBit equals 1 (retransmissions stopped), then the receiving entity shall
   a) Follow the rules in Section 8.4.1 on page 91.

4. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame and the RetransmissionDisableBit equals 0 (retransmissions started) then the receiving entity shall
   a) If the RetransmissionTimer is running, then stop it.
   b) If any unacknowledged I-frames have been sent then the RetransmissionTimer shall be restarted.
c) Follow the rules in Section 8.4.1 on page 91.

d) If the RetransmissionTimer is not running and the MonitorTimer is not running, then start the MonitorTimer.

On receipt of a valid S-frame or I-frame the ReqSeq contained in the frame shall acknowledge previously transmitted I-frames. ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with TxSeq up to and including (ReqSeq - 1) have been acknowledged.

8.4.5 Receiving REJ frames

Upon receipt of a valid REJ frame, where ReqSeq identifies an I-frame not yet acknowledged, the ReqSeq acknowledges I-frames with TxSeq up to and including ReqSeq - 1. Therefore the REJ acknowledges all I-frames before the I-frame it is rejecting.

ExpectedAckSeq shall be set equal to ReqSeq to mark I-frames up to and including ReqSeq - 1 as received.

NextTXSeq shall be set to ReqSeq to cause transmissions of I-frames to resume from the point where TxSeq equals ReqSeq.

If ReqSeq equals ExpectedAckSeq then the REJ frame shall be ignored.

8.4.6 Waiting acknowledgements

A counter, TransmitCounter, counts the number of times an L2CAP PDU has been transmitted. This shall be set to 1 after the first transmission. If the RetransmissionTimer expires the following actions shall be taken:

1. If the TransmitCounter is less than MaxTransmit then:
   a) Increment the TransmitCounter
   b) Retransmit the last unacknowledged I-frame, according to Section 8.4.1 on page 91.

2. If the TransmitCounter is equal to MaxTransmit this channel to the peer entity shall be assumed lost. The channel shall move to the CLOSED state and appropriate action shall be taken to report this to the upper layers.

8.4.7 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in Retransmission Mode are defined in this section.

8.4.7.1 TxSeq Sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.
The TxSeq sequence error may be due to three different causes:

- **Duplicated I-frame**
  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to ExpectedTxSeq – 1 (BufferSeq ≤ TxSeq < ExpectedTxSeq). The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4 on page 94. The Information field shall be discarded since it has already been received.

- **Out-of-sequence I-frame**
  The out-of-sequence I-frame is identified by a TxSeq within the legal range. The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4 on page 94.
  
  A REJ exception is triggered, and an REJ frame with ReqSeq equal to ExpectedTxSeq shall be sent to initiate recovery. The received I-frame shall be discarded.

- **Invalid TxSeq**
  An invalid TxSeq value is a value that does not meet either of the above conditions. An I-frame with an invalid TxSeq is likely to have errors in the control field and shall be silently discarded.

### 8.4.7.2 ReqSeq Sequence error

An ReqSeq sequence error exception condition occurs in the transmitter when a valid S-frame or I-frame is received which contains an invalid ReqSeq value. An invalid ReqSeq is one that is not in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

The L2CAP entity shall close the channel as a consequence of an ReqSeq Sequence error.

### 8.4.7.3 Timer recovery error

If an L2CAP entity fails to receive an acknowledgement for the last I-frame sent, then it will not detect an out-of-sequence exception condition and therefore will not transmit an REJ frame.

The L2CAP entity that transmitted an unacknowledged I-frame shall, on the expiry of the RetransmissionTimer, take appropriate recovery action as defined in Section 8.4.6 on page 95.

### 8.4.7.4 Invalid frame

Any frame received which is invalid (as defined in Section 3.3.6 on page 32) shall be discarded, and no action shall be taken as a result of that frame.
8.5 FLOW CONTROL MODE

When a link is configured to work in flow control mode, the flow control operation is similar to the procedures in retransmission mode, but all operations dealing with CRC errors in received packets are not used. Therefore

- REJ frames shall not be used in Flow Control Mode.
- The RetransmissionDisableBit shall always be set to zero in the transmitter, and shall be ignored in the receiver.

The behavior of flow control mode is specified in this section.

Assuming that the TxWindow size is equal to the buffer space available in the receiver (counted in number of I-frames), in flow control mode the number of unacknowledged frames in the transmitter window is always less than or equal to the number of frames for which space is available in the receiver. Note that a missing frame still occupies a place in the window.

![Figure 8.3: Overview of the receiver side when operating in flow control mode](image)

8.5.1 Transmitting I-frames

A new I-frame shall only be transmitted when the TxWindow is not full.

Upon transmission of the I-frame the following actions shall be performed:

- If no unacknowledged I-frames have been sent then the MonitorTimer shall be stopped and the RetransmissionTimer shall be started.
- If any I-frames have been sent and not acknowledged then the RetransmissionTimer remains active and no timer operation is performed.

The control field parameter ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTXSeq and NextTXSeq shall be incremented by one.
8.5.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be made available to the reassembly function. ExpectedTxSeq shall be incremented by one. An acknowledgement shall not be sent until the SDU reassembly function has pulled the I-frame.

Upon receipt of a valid I-frame with an out-of-sequence TxSeq (see Section 8.5.6 on page 99) all frames with a sequence number less than TxSeq shall be assumed lost and marked as missing. The missing I-frames are in the range from ExpectedTxSeq (the frame that the device was expecting to receive) up to TxSeq-1, (the frame that the device actually received). ExpectedTxSeq shall be set to TxSeq +1. The received I-frame shall be made available for pulling by the reassembly function. The acknowledgement shall not occur until the SDU reassembly function has pulled the I-frame. The ReqSeq shall be processed according to Section 8.5.4 on page 98.

8.5.3 I-frames pulled by the SDU reassembly function

When the L2CAP layer has removed one or more I-frames from the buffer, BufferSeq may be incremented in accordance with the amount of buffer space released. If BufferSeq is incremented, an acknowledgement shall be sent to the peer entity. If the MonitorTimer is active then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.

Note: Since the primary purpose of BufferSeq is to prevent buffer overflow, an implementation may choose to set BufferSeq in accordance with how many new incoming I-frames could be stored rather than how many have been removed.

The acknowledgement may be an RR or an I-frame. The acknowledgement shall be sent to the peer L2CAP entity with ReqSeq equal to BufferSeq. When there is no I-frame buffered for pulling, ExpectedTxSeq is equal to BufferSeq.

8.5.4 Sending and receiving acknowledgements

One of the timers MonitorTimer or RetransmissionTimer shall always be active while in Flow Control mode. Both timers shall never be active concurrently.

8.5.4.1 Sending acknowledgements

Whenever a data link layer entity transmits an I-frame or a S-frame, ReqSeq shall be set to ExpectedTxSeq or BufferSeq.

8.5.4.2 Receiving acknowledgements

On receipt of a valid S-frame or I-frame the ReqSeq contained in the frame shall be used to acknowledge previously transmitted I-frames. ReqSeq acknowledges I-frames with a TxSeq up to and including ReqSeq – 1.
1. If any outstanding I-frames were acknowledged then
   a) Stop the RetransmissionTimer
   b) If there are still unacknowledged I-frames then restart the
      RetransmissionTimer, otherwise start the MonitorTimer.
   c) Transmit any I-frames awaiting transmission according to Section
      8.5.1 on page 97.

   ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with
   TxSeq up to and including ExpectedAckSeq have been acknowledged.

8.5.5 Waiting acknowledgements

If the RetransmissionTimer expires the following actions shall be taken:
The I-frame supervised by the RetransmissionTimer shall be considered lost,
and ExpectedAckSeq shall be incremented by one.

1. If I-frames are waiting to be sent
   a) The RetransmissionTimer is restarted
   b) I-frames awaiting transmission are transmitted according to
      Section 8.5.1 on page 97.

2. If there are no I-frames waiting to be sent
   a) If there are still unacknowledged I-frames the RetransmissionTimer
      is restarted, otherwise the MonitorTimer is started.

8.5.6 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP
procedural errors. The error recovery procedures which are available following
the detection of an exception condition at the L2CAP layer in flow control only
mode are defined in this section.

8.5.6.1 TxSeq Sequence error

A TxSeq sequence error exception condition occurs in the receiver when a
valid I-frame is received which contains a TxSeq value which is not equal to the
expected value, thus TxSeq is not equal to ExpectedTxSeq.

The TxSeq sequence error may be due to three different causes:
• *Duplicated I-frame*
  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to
  ExpectedTxSeq – 1. The ReqSeq shall be processed according to Section
  8.5.4 on page 98. The Information field shall be discarded since it has
  already been received.
• **Out-of-sequence I-frame**

The out-of-sequence I-frame is identified by a TxSeq within the legal range $\text{ExpectedTxSeq} < \text{TxSeq} < (\text{BufferSeq} + \text{TxWindow})$. The ReqSeq shall be processed according to Section 8.5.4 on page 98.

The missing I-frame(s) are considered lost and ExpectedTXSeq is set equal to TxSeq+1 as specified in Section 8.5.2 on page 98. The missing I-frame(s) are reported as lost to the SDU reassembly function.

• **Invalid TxSeq**

An invalid TxSeq value is a value that does not meet either of the above conditions and TxSeq is not equal to ExpectedTxSeq. An I-frame with an invalid TxSeq is likely to have errors in the control field and shall be silently discarded.

**8.5.6.2 ReqSeq Sequence error**

An ReqSeq sequence error exception condition occurs in the transmitter when a valid S-frame or I-frame is received which contains an invalid ReqSeq value. An invalid ReqSeq is one that is not in the range $\text{ExpectedAckSeq} \leq \text{ReqSeq} \leq \text{NextTXSeq}$.

The L2CAP entity shall close the channel as a consequence of an ReqSeq Sequence error.

An L2CAP entity that fails to receive an acknowledgement for an I-frame shall, on the expiry of the RetransmissionTimer, take appropriate recovery action as defined in Section 8.5.5 on page 99.

**8.5.6.3 Invalid frame**

Any frame received that is invalid (as defined in Section 3.3.6 on page 32) shall be discarded, and no action shall be taken as a result of that frame, unless the receiving L2CAP entity is configured to deliver erroneous frames to the layer above L2CAP. In that case, the data contained in invalid frames may also be added to the receive buffer and made available for pulling from the SDU reassembly function.

**8.6 ENHANCED RETRANSMISSION MODE**

Enhanced Retransmission mode operates as an HDLC balanced data link operational mode. Either L2CAP entity may send frames at any time without receiving explicit permission from the other L2CAP entity. A transmission may contain single or multiple frames and shall be used for I-frame transfer and/or to indicate status change.
8.6.1 Function Of PDU Types

Enhanced Retransmission mode uses I-frames to transfer upper layer information and S-frames for supervision. There are four S-frames defined: Receiver Ready (RR), Reject (REJ), Receiver Not Ready (RNR), and Selective Reject (SREJ). All frames formats in Enhanced Retransmission mode shall use the Enhanced Control Field.

8.6.1.1 Receiver Ready (RR)

The RR frame shall be used by an L2CAP entity to
1. Indicate that it is ready to receive I-frames
2. Acknowledge previously received I-frames numbered up to ReqSeq - 1 inclusive.

An RR with P-bit set to 1 (P=1) is used to indicate the clearance of any busy condition that was initiated by an earlier transmission of an RNR frame by the same L2CAP entity.

8.6.1.2 Reject (REJ)

The REJ frame shall be used by an L2CAP entity to request retransmission of I-frames starting with the frame numbered ReqSeq. I-frames numbered ReqSeq - 1 and below shall be considered acknowledged. Additional I-frames awaiting initial transmission may be transmitted following the retransmitted I-frame(s) up to the TxWindow size of the receiver.

At most only one REJ exception from a given L2CAP entity to another L2CAP entity shall be established at any given time. A REJ frame shall not be transmitted until an earlier REJ exception condition or all earlier SREJ exception conditions have been cleared. The REJ exception condition shall be cleared upon the receipt of an I-frame with TxSeq equal to the ReqSeq of the REJ frame.

Two L2CAP entities may be in REJ exception conditions with each other at the same time.

8.6.1.3 Receiver Not Ready (RNR)

The RNR frame shall be used by an L2CAP entity to indicate a busy condition (i.e. temporary inability to receive I-frames). I-frames numbered up to ReqSeq - 1 inclusive shall be considered acknowledged. The I-frame numbered ReqSeq and any subsequent I-frames sent shall not be considered acknowledged. The acceptance status of these I-frames shall be indicated in subsequent transfers.
8.6.1.4 Selective Reject (SREJ)

The SREJ frame shall be used by an L2CAP entity to request retransmission of one I-frame. The ReqSeq shall indicate the TxSeq of the earliest I-frame to be retransmitted (not yet reported by a SREJ). If the P-bit is set to 1 then I-frames numbered up to ReqSeq -1 inclusive shall be considered acknowledged. If the P-bit is set to 0 then the ReqSeq field in the SREJ shall not indicate acknowledgement of I-frames.

Each SREJ exception condition shall be cleared upon receipt of an I-frame with TxSeq equal to the ReqSeq sent in the SREJ frame.

An L2CAP entity may transmit one or more SREJ frames with the P=0 before one or more earlier SREJ exception conditions initiated with SREJ(P=0) have been cleared. An L2CAP entity shall not transmit more than one SREJ with P=1 before all earlier SREJ exception conditions have been cleared. A SREJ frame shall not be transmitted if an earlier REJ exception condition has not been cleared. Likewise a REJ frame shall not be transmitted if one or more SREJ exception conditions have not been cleared. Only one I-frame shall be retransmitted in response to receiving a SREJ frame with P=0. Additional I-frames awaiting initial transmission may be transmitted following the retransmission of the specific I-frame requested by SREJ with P=1.

8.6.1.5 Functions of the Poll (P) and Final (F) bits.

P-bit set to 1 shall be used to solicit a response frame with the F-bit set to 1 from the remote L2CAP entity at the earliest respond opportunity. At most only one frame with a P=1 shall be outstanding in a given direction at a given time. Before an L2CAP entity issues another frame with P=1, it shall have received a response frame from the remote L2CAP entity with F=1. If no valid frame is received with F=1 within Monitor time-out period, the frame with P=1 may be retransmitted.

The Final bit shall be used to indicate the frame as a response to a soliciting poll (S-frame with P=1). The frame with F=1 shall not be retransmitted. The Monitor-timeout is not used to monitor lost frames with F=1. Additional frames with F=0 may be transmitted following the frame with F=1.

S-frames shall not be transmitted with both the F-bit and the P-bit set to 1 at the same time.

8.6.2 Rules For Timers

Timers are started upon transmission of a packet. Timers should be started when the corresponding packet leaves the controller (transmitted or flushed). If the timer is not started when the packet leaves the controller then it shall be started when the packet is delivered to the controller. If a flush timeout does not exist on the link for the channel using Enhanced Retransmission mode then the value for the Retransmission time-out shall be at least two seconds and the
value for the Monitor time-out shall be at least twelve seconds. If a flush timeout exists on the link for Enhanced Retransmission mode then the value for the Retransmission time-out shall be the larger of:

- The value of the flush timeout multiplied by three
- one second

If a flush timeout exists on the link for Retransmission mode and both sides of the link are configured to the same flush timeout value then the monitor timeout shall be set to a value at least as large as the Retransmission time-out otherwise the value of the Monitor time-out shall be the larger of:

- the value of the flush timeout multiplied by six
- twelve seconds

If an L2CAP entity knows that a specific packet has been flushed instead of transmitted then it may execute proper error recovery procedures immediately.

**Note:** If the link has a flush timeout and the Non-Flushable Packet Boundary Flag feature is used to mark the Enhanced Retransmission mode packets as non-flushable then the link does not have a flush timeout with regards to Enhanced Retransmission mode.

### 8.6.3 General Rules for the State Machine

Enhanced Retransmission mode is specified using a pair of state machines, a Transmitter state machine and a Receiver state machine. The following rules apply to the state machine pair.

1. The state machine pair is informative but described using normative text in order to clearly specify the behavior of the protocol. Designers and implementers may choose any design / implementation technique they wish, but it shall behave in a manner identical to the external behavior of the specified state machines.

2. There is a single state machine pair for each active L2CAP channel configured to use Enhanced Retransmission mode.

3. Variables are used to limit the number of states by maintaining the state of particular conditions. The variables are defined in section 8.6.5.2.

4. For some combinations of Event and Condition, the state tables provide alternative groups of actions. These alternatives are separated by horizontal lines in the Actions and Next State columns. The alternatives are mutually exclusive; selection of an alternate is done based upon (i) local status, (ii) a layer management action, or (iii) an implementation decision. There is no relationship between the order of the alternatives between events, nor is it implied that the same alternative must be selected every time the event occurs.

5. The state tables use timers. Any Start Timer action restarts the specified timer from its initial value, even if the timer is already running. When the
timer reaches 0 the appropriate timer expired event is set and the timer stops. The Stop Timer action stops a timer if it is running.

6. Events not recognized in a particular state are assumed to remain pending until any masking flag is modified or a transition is made to a state where they can be recognized.

7. Some state transitions and actions are triggered by internal events (e.g. requests from the upper layer). It is implementation specific how these internal events are realized. They are used for clarity in specifying the state machine. All events including Internal events are described in Section 8.6.5.3 on page 106.

8. The state machines specify the exact frames to be sent by transmitters but are relaxed on what receivers are allowed to accept as valid. For example there are cases where the transmitter is required to send a frame with P=1. The correct response is a frame with F=1 but in some cases the receiver is allowed to accept a frame with F=0 in addition to F=1.

8.6.4 State Diagram

The state diagram shows the states and the main transitions. Not all events are shown on the state diagram.
8.6.5 States Tables

8.6.5.1 State Machines

Enhanced Retransmission mode is described as a pair of state machine. The Receiver state machine handles all received frames while the Transmitter State machine handles all asynchronous events including requests from the upper layer and the expiration of timers.

The Receiver state machine "calls" the Transmitter state machine using the PassToTx action. This shows up in the Transmitter state machine as an event. When the Transmitter state machine is called it runs to completion before returning to the Receiver state machine. Running to completion means that all actions are executed and the Transmitter state is changed to the new state.

The Receiver and Transmitter state machine share variables and timers.
8.6.5.2 States

The following states have been defined to specify the protocol; the actual number of states and naming in a given implementation is outside the scope of this specification:

RECV—This is the main state of the Receiver state machine.

REJ_SENT—The L2CAP entity has sent a REJ frame to cause the remote L2CAP entity to resend l-frame(s). The L2CAP entity is waiting for the l-frame with a TxSeq that matches the ReqSeq sent in the REJ. Whether to send a REJ versus a SREJ is implementation dependent.

SREJ_SENT—The L2CAP entity has sent one or more SREJ frames to cause the remote L2CAP entity to resend missing l-frame(s). The local L2CAP entity is waiting for all requested l-frames to be received. If additional missing l-frames are detected while in SREJ_SENT then additional SREJ frames or a REJ frame can be sent to request those l-frames. Whether to send a SREJ versus a REJ is implementation dependent.

XMIT—This is the main state of the Transmitter state machine.

WAIT_ACK—The Retransmission timer has expired while in the XMIT state and an RR (P=1) or RNR (P=1) has been sent to solicit the current state of received frames from the remote L2CAP entity (i.e. next TxSeq expected to be received). The L2CAP entity is waiting for a frame with F=1.

WAIT_F—Local Busy condition has cleared and an RR P=1 has been sent to alert the remote device while in the XMIT state. The L2CAP entity is waiting for a frame with F=1. RR P=1 is used so that polling by the remote station is not required when a station is in local busy. This state is similar to WAIT_ACK except that l-frames can be sent while in WAIT_F.

8.6.5.3 Variables and Timers

Variables are used to limit the number of states and help clarify the state chart tables. Variables can be set to values, evaluated in conditions and compared in conditional statements. They are also used in the action descriptions. Below is a list of the operators, connectives and statements that can be used with variables.

<table>
<thead>
<tr>
<th>Operator, connective or statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:=</td>
<td>Assignment operator. Used to set a variable to a value</td>
</tr>
<tr>
<td>=</td>
<td>Relational operator &quot;equal&quot;</td>
</tr>
</tbody>
</table>

Table 8.2: Enhanced Retransmission mode uses the following variables and sequence numbers described in Section 8.3 on page 86
<table>
<thead>
<tr>
<th>Operator, connective or statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Relational operator &quot;greater than&quot;.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Relational operator &quot;less than&quot;</td>
</tr>
<tr>
<td>≥</td>
<td>Relational operator &quot;greater than or equal&quot;</td>
</tr>
<tr>
<td>≤</td>
<td>Relational operator &quot;less than or equal&quot;</td>
</tr>
<tr>
<td>+</td>
<td>Arithmetic operator &quot;plus&quot;</td>
</tr>
<tr>
<td>mod</td>
<td>Modulo operator - returns the remainder of division of one number by another.</td>
</tr>
<tr>
<td>and</td>
<td>logical connective &quot;and&quot;. It returns TRUE if both operands are TRUE otherwise it returns FALSE.</td>
</tr>
<tr>
<td>or</td>
<td>logical connective &quot;or&quot;. It returns TRUE if either of its operands are TRUE otherwise it returns FALSE.</td>
</tr>
<tr>
<td>if (expression) then statement</td>
<td>Conditional Statement. If expression is TRUE then the statement is executed otherwise the statement is not executed. The statement is composed of one or more actions. All the actions in the statement are indented under the if … then clause.</td>
</tr>
<tr>
<td>if (expression) then statement1</td>
<td>Conditional Statement. If expression is TRUE then statement1 is executed otherwise statement2 is executed. A statement is composed of one or more actions. All the actions in the statement1 are indented under the if … then clause and all the actions of statement2 are indented under the else clause.</td>
</tr>
<tr>
<td>else statement2</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: Enhanced Retransmission mode uses the following variables and sequence numbers described in Section 8.3 on page 86

- NextTxSeq
- ExpectedAckSeq
- ReqSeq
- ExpectedTxSeq
- BufferSeq
- In addition to the variables above the following variables and timers are used.

**RemoteBusy**—when set to TRUE RemoteBusy indicates that the local L2CAP entity has received an RNR from the remote L2CAP entity and considers the remote L2CAP entity as busy. When the remote device is busy it will likely discard I-frames sent to it. The RemoteBusy flag is set to FALSE when the local L2CAP Entity receives an RR, REJ or SREJ. When set to FALSE the local L2CAP entity considers the remote L2CAP entity able to accept I-frames. When the channel is created RemoteBusy shall be set to FALSE.
**LocalBusy**—when set to TRUE, LocalBusy indicates the local L2CAP entity is busy and will discard received I-frames. When set to FALSE the local L2CAP entity is not busy and is able to receive I-frames. When the channel is created LocalBusy shall be set to FALSE.

**UnackedFrames**—holds the number of unacknowledged I-frames. When the channel is created UnackedFrames shall be set to 0.

**UnackedList**—holds the unacknowledged I-frames so they can be retransmitted if necessary. I-frames in the list are accessed via their TxSeq number. For example UnackedList[5] accesses the I-frame with TxSeq 5.

**PendingFrames**—holds the number of pending I-frames. I-frames passed to L2CAP from the upper layer may not be able to be sent immediately because the remote L2CAP entity’s TxWindow is full, is in a busy condition or the local L2CAP is in the incorrect state. When I-frames cannot be sent they are stored in a queue until conditions allow them to be sent. When the channel is created PendingFrames shall be set to 0.

**SrejList**—is a list of TxSeq values for I-frames that are missing and need to be retransmitted using SREJ. A SREJ has already been sent for each TxSeq on the list. When SrejList is empty it equals 0 (i.e. SrejList = 0). If SrejList is not empty it is greater than 0 (i.e. SrejList > 0).

**RetryCount**—holds the number of times an S-frame operation is retried. If an operation is tried MaxTransmit times without success the channel shall be closed.

**RetryIFrames[]**—holds a retry counter for each I-frame that is sent within the receiving device’s TxWindow. Each time an I-frame is retransmitted the corresponding counter within RetryIFrames is incremented. When an attempt to retransmit the I-frame is made and the counter is equal to MaxTransmit then the channel shall be closed.

**RNRsent**—when set to TRUE it means that the local L2CAP entity has sent an RNR frame. It is used to determine if the L2CAP entity needs to send an RR to the remote L2CAP entity to clear the busy condition. When the channel is created RNRsent shall be set to FALSE.

**RejActioned**—is used to prohibit a frame with F=1 from causing I-frames already retransmitted in response to a REJ from being retransmitted again. RejActioned is set to TRUE if a received REJ is actioned when a frame sent with P=1 is unanswered. When the channel is created RejActioned shall be set to FALSE.

**SrejActioned**—is used in conjunction with SrejSaveReqSeq to prohibit a frame with F=1 from causing an I-frame already retransmitted in response to a SREJ from being retransmitted again. SrejActioned is set to TRUE if a received SREJ is actioned when a frame sent with P=1 is unanswered. When the channel is created SrejActioned shall be set to FALSE.
**SrejSaveReqSeq**—is used to save the ReqSeq of a SREJ frame that causes SrejActioned to be set to TRUE.

**SendRej**—when set to TRUE it indicates that the local L2CAP entity has determined that a REJ should be sent in the SREJ_SENT state while processing received I-frames. The sending of new SREJ frames is stopped. When the channel is created SendRej shall be set to FALSE.

**BufferSeqSrej**—is used while in the SREJ_SENT state to keep track of the value to which BufferSeq will be set upon exit of the SREJ_SENT state.

**FramesSent**—is used to keep track of the number I-frames sent by the SendData and Retransmit-I-frames actions.

**RetransTimer**—The Retransmission Timer is used to detect lost I-frames. When the channel is created the RetransTimer shall be off.

**MonitorTimer**—The Monitor Timer is used to detect lost S-frames. When the channel is created the MonitorTimer shall be off.

### 8.6.5.4 Events

**Data-Request**—The upper layer has requested that an SDU be sent. The SDU may need to be broken into multiple I-frames by L2CAP based on the MPS of the remote device and/or the maximum PDU allowed by the HCI or QoS requirements of the system.

**Local-Busy-Detected**—A local busy condition occurs when the local L2CAP entity is temporarily unable to receive, or unable to continue to receive, I-frames due to internal constraints. For example, the upper layer has not pulled received I-frames and the local L2CAP entity needs to send an acknowledgment to the remote L2CAP entity. The method for handling the detection of local busy is implementation specific. An implementation may wait to send an RNR to see if the busy condition will clear before the remote L2CAP entity’s Retransmission timer expires. If the busy condition clears then frames can be acknowledged with an RR or I-frame. If the busy condition does not clear before the remote L2CAP entity’s Retransmission timer expires then an RNR shall be sent in response to the RR or RNR poll sent by the remote L2CAP entity. Optionally an implementation may send an RNR as soon as the local busy condition is detected.

**Local-Busy-Clear**—The local busy condition clears when L2CAP has buffer space to receive more I-frames (i.e. SDU Reassembly function and/or upper layer has pulled I-frames) and if necessary the upper layer has cleared the busy condition.

**Recv ReqSeqAndFbit**—This is an event generated by the Receiver state machine. It contains the ReqSeq and F-bit value of a received frame. The value of the F-bit can be checked in a condition.
**Recv Fbit**—This is an event generated by the Receiver state machine. It contains the F-bit value of a received frame. The value of the F-bit can be checked in a condition.

**RetransTimer-Expires**—The Retransmission Timer has counted to down to 0 and stopped.

**MonitorTimer-Expires**—The Monitor Timer has counted down to 0 and stopped.

**Recv I-frame**—Receive an I-frame with any value for the F-bit.

**Recv RR, REJ, RNR, SREJ (P=x) or (F=x)**—Receive a specific S-frame (RR, REJ, etc.) with a specific value for the P and/or F bit. The F-bit and the P-bit shall not both be set to 1 in a transmitted S-frame so received S-frames with both P and F set to 1 should be ignored. If the P and/or F bit value is not specified in the event then either value is accepted.

**Recv RRorRNR**—Receive an RR or RNR with any value for the P-bit and F-bit.

**Recv REJorSREJ**—Receive an REJ or SREJ with any value for the P-bit and F-bit.

**Recv frame**—This is catch-all for all frames that are not explicitly declared as events in the state table.

### 8.6.5.5 Conditions

**RemoteBusy = TRUE or FALSE**—TRUE indicates the remote L2CAP entity is in a busy condition and FALSE indicates the remote L2CAP entity is not busy.

**LocalBusy = TRUE or FALSE**—TRUE indicates the local L2CAP entity is in a busy condition and FALSE indicates the local L2CAP entity is not busy.

**RemWindow-Not-Full**—The number of unacknowledged I-frames sent by L2CAP has not yet reached the TxWindow size of the remote L2CAP entity.

**RemWindow-Full**—The number of unacknowledged I-frames sent by the L2CAP entity has reached the TxWindow size of the remote L2CAP entity. No more I-frames shall be sent until one or more I-frames have been acknowledged.

**RNRsent = TRUE or FALSE**—TRUE indicates an RNR has been sent while a local busy condition exists. It is set to FALSE when the local busy condition clears.

**F = 0 or 1**—the F-bit of a received frame is checked. The F-bit of the received frame is available as part of the Recv ReqSeqAndFbit and Recv Fbit events.
RetryIframes[i] < or \geq \text{MaxTransmit} — Compare the appropriate counter in RetryIframes after processing the ReqSeq in the receive frame to determine if it has reached MaxTransmit or not.

RetryCount < or \geq \text{MaxTransmit} — Compare RetryCount to determine if it has reached MaxTransmit or not.

With-Expected-TxSeq — The TxSeq of a received I-frame is equal to ExpectedTxSeq.

With-Valid-ReqSeq — The ReqSeq of the received frame is in the range .

With-Valid-ReqSeq-Retrans — The ReqSeq of the received frame is in the range ExpectedAckSeq \leq \text{ReqSeq} < \text{NextTxSeq}.

With-Valid-F-bit — The F-bit of a received frame is valid if it is 0 or if it is 1 and a frame sent with P=1 by the local L2CAP entity is unanswered (i.e. the local L2CAP entity send a frame with P=1 and has not yet received a frame with F=1 until receiving this one). If the Transmitter state machine is in the WAIT_ACK or WAIT_F states then a frame sent with P=1 is unanswered.

With-unexpected-TxSeq — The TxSeq of the received I-frame is within the TxWindow of the L2CAP entity receiving the I-frame but has a TxSeq "greater" than ExpectedTxSeq where "greater" means later in sequence than ExpectedTxSeq.

With-duplicate-TxSeq — The TxSeq of the received I-frame is within the TxWindow of the L2CAP entity receiving the I-frame but has a TxSeq "less" than ExpectedTxSeq where "less" means earlier in the sequence than ExpectedTxSeq. In other words this is a frame that has already been received.

With-Invalid-TxSeq — The TxSeq of the received I-frame is not within the TxWindow of the L2CAP entity receiving the frame.

With-Invalid-ReqSeq — The ReqSeq of the received frame is not in the range ExpectedAckSeq \leq \text{ReqSeq} < \text{NextTxSeq}.

With-Invalid-ReqSeq-Retrans — The ReqSeq of the received frame is not in the range ExpectedAckSeq \leq \text{ReqSeq} < \text{NextTxSeq}.

Not-With-Expected-TxSeq — The TxSeq of the received I-frame is within the TxWindow of the L2CAP entity receiving the frame but is not equal to ExpectedTxSeq. It is either unexpected or a duplicate.

With-Expected-TxSeq-Srej — The TxSeq of the received I-frame is equal to the TxSeq at the head of SrejList.

SendRej = TRUE or FALSE — TRUE indicates that a REJ will be sent after all frames requested using SREJ have been received.
SrejList = or > 1—Determine if the number of items in SrejList is equal to or greater than 1.

With-Unexpected-TxSeq-Srej—The TxSeq of the received I-frame is equal to one of the values stored in SrejList but is not the TxSeq at the head. This indicates that one or more I-frames requested using SREJ are missing either because the SREJ was lost or the requested I-frame(s) were lost. Either way the SREJ frames must be resent to retrieve the missing I-frames.

With-duplicate-TxSeq-Srej—The TxSeq of the received I-frame is equal to a TxSeq of one of the saved I-frames indicating it is a duplicate.

8.6.5.6 Actions

Send-Data—This action is executed as a result of a Data-Request event. The number of I-frames sent without being acknowledged shall not exceed the TxWindow size of the receiving L2CAP entity (UnackedFrames is less than or equal to the remote L2CAP entity’s TxWindow). Any I-frames that cannot be sent because they would exceed the TxWindow size are queued for later transmission. For each I-frame the following actions shall be carried out:

- Send I-frame with TxSeq set to NextTxSeq and ReqSeq set to BufferSeq.
- UnackedList[NextTxSeq] := I-frame
- UnackedFrames := UnackedFrames + 1
- FramesSent := FramesSent + 1
- RetryIframes[NextTxSeq] := 1
- NextTxSeq := (NextTxSeq + 1) mod 64
- If the RetransTimer is not already running then Start-RetransTimer

Pend-Data—This action is executed as a result of a Data-Request when it is not possible to send I-frames because the window is full, the remote L2CAP entity is in a busy condition or the local L2CAP entity is not in a state where I-frames can be sent (e.g. WAIT_ACK). The I-frame(s) are queued for later transmission.

Process-ReqSeq—the ReqSeq contained in the received frame shall acknowledge previously transmitted I-frames. ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with TxSeq up to and including (ReqSeq—1) have been acknowledged. The acknowledged I-frames shall be removed from UnackedList, the retry counters for each acknowledged frame shall be set to 0 and the number of acknowledged frames shall be subtracted from UnackedFrames so that UnackedFrames shall contain the number of the remaining unacknowledged I-frames. Pending I-frames are now available to be transmitted by the Send-Ack action.

Send RR, RNR (P=x) or (F=x)—Send the specified S-frame with the specified value for the P-bit or F-bit. If a value for the P-bit or F-bit is not specified the
value shall be 0. For example Send RR(P=1) means send an RR with the P-bit set to 1 and the F-bit set to 0. The ReqSeq field shall be set to BufferSeq.

Send REJ (P=x) or (F=x)—Send a REJ with the specified value for the P-bit or F-bit. The ReqSeq field shall be set to ExpectedTxSeq. If a value for the P-bit or F-bit is not specified the value shall be 0. Note that this will acknowledge previously received I-frames up to ExpectedTxSeq—1 and may allow the remote L2CAP entity to transmit new I-frames. If the local L2CAP entity is not in a position to acknowledge the previously received I-frames it may use SREJ(P=0) or RNR. It may also wait to send the REJ until it is able to acknowledge the I-frames.

Send RRorRNR (P=x) or (F=x)—Send an RR or RNR with the specified value for the P-bit or F-bit based on the value of LocalBusy. If a value for the P-bit or F-bit is not specified the value shall be 0. An RNR shall be sent if LocalBusy equals TRUE. If LocalBusy equals FALSE then an RR shall be sent.

Send IorRRorRNR(F=x)—Send I-frames, an RR or an RNR with the specified value for the F-bit. If a value for the F-bit is not specified the value shall be 0. The following algorithm shall be used:

\[
\begin{align*}
\text{FramesSent} & := 0 \\
& \text{if RemoteBusy = TRUE then} \\
& \quad \text{Retransmit-I-Frames} \\
& \quad \text{Send-Pending-I-frames (see note)} \\
& \text{if LocalBusy = TRUE then} \\
& \quad \text{Send RNR (see note)} \\
& \text{else if FramesSent := 0 then} \\
& \quad \text{Send RR(f=x)}
\end{align*}
\]

Note: The SendIorRRorRNR(f=x) sends frames by invoking other actions. During the execution of SendIorRRorRNR multiple actions may be invoked. The first action invoked shall send the first or only frame with the F-bit set as passed into the SendIorRRorRNR action. All other frames sent shall have the F-bit set to 0.

Send SREJ—Send one or more SREJ frames with P=0. For each missing I-frame starting with ExptectedTxSeq up to but not including the TxSeq of the received I-frame, an SREJ frame is sent with ReqSeq set to the TxSeq of the missing frame. The TxSeq is inserted into the tail of SrejList. For example if ExpectedTxSeq is 3 and the received I-frame has a TxSeq of 5 there are two missing I-frames. An SREJ with ReqSeq 3 is sent followed by an SREJ with ReqSeq 4. TxSeq 3 is inserted first into SrejList followed by TxSeq 4. After all SREJ frames have been sent ExpectedTxSeq shall be set to the TxSeq of the received I-frame + 1 mod 64.

Send SREJ(SrejList)—Send one or more SREJ frames with P=0. An I-frame was received that matches one of the TxSeq values in the SrejList but does not
match the head of SrejList. This means I-frames requested via SREJ are still missing. For each TxSeq value starting with the head of SrejList and going backwards (from the head towards the tail) through the SrejList up to but not including the TxSeq of the received frame, an SREJ frame is sent with ReqSeq set to the TxSeq from SrejList. The TxSeq is removed from SrejList and reinserted into the tail of SrejList.

**Send SREJ(SrejList-tail)(F=1)**—Send a SREJ frame with F=1 and ReqSeq equal to the TxSeq at the tail of SrejList.

**Start-RetransTimer**—If the Monitor timer is not running then start the Retransmission Timer from its initial value (see Retransmission time-out in Section 5.4). If the Retransmission timer is already running it is restarted from its initial value. If the Monitor timer is running then the Retransmission timer is not started.

**Start-MonitorTimer**—Start the Monitor Timer from its initial value (see Monitor time-out in Section 5.4 on page 57). If the timer is already running it is restarted from its initial value.

**PassToTx**—Pass the ReqSeq and F-bit value of a received frame to the Transmitter state machine. This will show up as a Recv ReqSeqAndFbit event in the Transmitter state machine.

**PassToFbit**—Pass the F-bit value of a received frame to the Transmitter state machine. This will show up as a Recv Fbit event in the Transmitter state machine.

**Data-Indication**—A received I-frame is passed to the SDU reassembly function. For the purpose of the state machine this operation is completed immediately so the Send_Ack action should be executed as one of the next actions. In some cases the SDU reassembly function cannot accept the I-frame so the I-frame will be stored within the L2CAP Entity consuming a portion of its TxWindow. When the I-frame is pulled by the SDU reassembly function the Send_Ack action should be executed. Before the Send_Ack action is executed BufferSeq is advanced as follows:

\[
\text{BufferSeq} := (\text{BufferSeq} + 1) \mod 64
\]

**Increment-ExpectedTxSeq**—ExpectedTxSeq is incremented as follows:

\[
\text{ExpectedTxSeq} := (\text{ExpectedTxSeq} + 1) \mod 64
\]

**Stop-RetransTimer**—the Retransmission timer is stopped.

**Stop-MonitorTimer**—the Monitor timer is stopped.

**Send-Ack (F=x)**—an acknowledgement with the specified value for the F-bit may be sent. Note that this action may occur in an action block with other actions that also send frames. If a frame has already been sent then it is not necessary to send additional frames. If the value for the F-bit is not specified it shall be set to 0. If the value specified is P then the F-bit shall be set equal to the value of the P-bit of the received frame being acknowledged. If more than
one frame is sent in the acknowledgment only the first frame shall have an F-bit set to 1. An acknowledgement is an RR, RNR, or pending I-frame(s) (I-frames that have not been transmitted yet). If pending I-frames are available and are allowed to be sent then as many as allowed should be sent as an acknowledgment. Sending an RR or RNR as an acknowledgment for each received I-frame is not required. An implementation may wait to send an RR or RNR until a specific number of I-frames have been received, after a certain period of time has elapsed or some other algorithm. To keep data flowing it is recommended that an acknowledgment be sent before the TxWindow is full. It should also be noted that the maximum size of a remote L2CAP entity’s unacknowledged I-frame list may be smaller than the local L2CAP entity’s TxWindow. Therefore the local L2CAP entity should not expect the remote L2CAP entity to send enough frames to fill its TxWindow and should acknowledge I-frames accordingly. The following algorithm shall be used when sending an acknowledgment.

\[
\begin{align*}
\text{if} & \quad \text{LocalBusy == TRUE then} \\
& \quad \text{Send\_RNR}(F=x) \\
\text{else if} & \quad \text{RemoteBusy == FALSE and Pending I-frames Exist and RemWindow-Not-Full} \\
& \quad \text{Send\_Pending-I-frames} (F=x) \\
\text{else} & \quad \text{Send\_RR} (F=x)
\end{align*}
\]

**InitSrej**—Initialize the variables used for processing SREJ as follows:

- **Clear SrejList**—(remove all values)
- **SendRej** := FALSE
- **BufferSeqSrej** := **BufferSeq**

**SaveIframeSrej**—Save the received I-frame. Missing I-frame(s) will be retransmitted in response to SREJ frames. Implementations may want to save the I-frame in its proper sequence order by leaving room for the missing I-frames.

**StoreOrIgnore**—If the local L2CAP entity has room to store the received I-frame then it may store it otherwise it shall discard it.

**PbitOutstanding**—If the Transmitter state machine of the local L2CAP entity is in the WAIT\_ACK state or the WAIT\_F state then return TRUE otherwise return FALSE.

**Retransmit-I-frames**—All the unacknowledged I-frames starting with the I-frame with TxSeq equal to the ReqSeq field of the received S-frame (REJ or RR) is retransmitted. If the P-bit of the received S-frame is 1 then the F-bit of the first I-frame sent shall be 1. If the P-bit of the received S-frame is 0 then the F-bit of the first I-frame sent shall be 0. The F-bit of all other unacknowledged I-frames sent shall be 0. The retry counter in RetryIframes[] for each retransmitted I-frame is incremented by 1. FramesSent shall be incremented by 1 for
each frame sent. If the RetransTimer is not already running then perform the Start-RetransTimer action.

**Retransmit-Requested I-frame**—The unacknowledged I-frame with TxSeq equal to the ReqSeq field of the received S-frame (SREJ) is retransmitted. If the P-bit of the received S-frame is 1 then the F-bit of the retransmitted I-frame shall be 1. If the P-bit of the received S-frame is 0 then the F-bit of the retransmitted I-frame shall be 0. The retry counter in RetryIframes[] corresponding to the retransmitted I-frame is incremented by 1. If the RetransTimer is not already running then perform the Start-RetransTimer action.

**Send-Pending I-frames (F=x)**—send all pending I-frames that can be sent without exceeding the receiver’s TxWindow using the Send-Data action. If a value for the F-bit is specified then the F-bit of the first I-frame sent shall be set to the specified value and the F-bit of all other I-frames sent shall be set to 0. If no value for the F-bit is specified then all I-frames sent shall have the F-bit set to 0. Pending I-frames are I-frames that have been given to the L2CAP entity by the upper layer but have not yet been transmitted. If one or more I-frames are sent and the RetransTimer is not already running then perform the Start-RetransTimer action.

**Close Channel**—Close the L2CAP channel as described in section 4.6.

**Ignore**—the event may be silently discarded.

**PopSrejList(TxSeq)**—Remove and discard the TxSeq from the head of SrejList.

**Data-IndicationSrej**—If the received I-frame fills a gap in a sequence of saved I-frames then all the saved I-frames in the sequence are passed to the SDU reassembly function. For the purpose of the state machine this operation is completed immediately. For example if the TxSeq of saved I-frames before receiving an I-frame is 2, 3, 5, 6, 9 and the received I-frame has a TxSeq of 4 then it fills the gap between 3 and 5 so the sequence 2, 3, 4, 5, 6 can be passed to the SDU reassembly function. When the I-frames are actually removed from the L2CAP entity receive buffers either by being processed immediately or when pulled by the SDU reassembly function, BufferSeqSrej is advanced as follows:

\[ \text{BufferSeqSrej} := (\text{BufferSeqSrej} + 1) \mod 64 \]
## 8.6.5.7 XMIT State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-Request</td>
<td>RemoteBusy = FALSE and RemWindow-Not-Full</td>
<td>Send-Data</td>
<td>XMIT</td>
</tr>
<tr>
<td>Data-Request</td>
<td>RemoteBusy = TRUE or RemWindow-Full</td>
<td>Pend-Data</td>
<td>XMIT</td>
</tr>
<tr>
<td>Local-Busy-Detected</td>
<td></td>
<td>LocalBusy := TRUE</td>
<td>XMIT</td>
</tr>
<tr>
<td>Local-Busy-Clear</td>
<td>RNRsent = TRUE</td>
<td>LocalBusy := TRUE</td>
<td>WAIT_F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RNR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RNRsent := TRUE;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RR(P=1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RetryCount := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop-RetransTimer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-MonitorTimer</td>
<td></td>
</tr>
<tr>
<td>Local-Busy-Clear</td>
<td>RNRsent = FALSE</td>
<td>LocalBusy := FALSE</td>
<td>XMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RNRsent = FALSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RRorRNR(P=1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RetryCount := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-MonitorTimer</td>
<td></td>
</tr>
<tr>
<td>Recv ReqSeqAndFbit</td>
<td></td>
<td>Process-ReqSeq</td>
<td>XMIT</td>
</tr>
<tr>
<td>Recv Fbit</td>
<td></td>
<td></td>
<td>XMIT</td>
</tr>
<tr>
<td>RetransTimer-Expires</td>
<td></td>
<td>Send RRorRNR(P=1)</td>
<td>WAIT_ACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RetryCount := 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-MonitorTimer</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.3: XMIT state table
### 8.6.5.8 WAIT_ACK State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-Request</td>
<td></td>
<td>Pend-Data</td>
<td>WAIT_ACK</td>
</tr>
<tr>
<td>Recv ReqSeq-ReqAndFbit</td>
<td>F = 1</td>
<td>Process-ReqSeq</td>
<td>XMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop-MonitorTimer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If UnackedFrames &gt; 0 then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-RetransTimer</td>
<td></td>
</tr>
<tr>
<td>Recv ReqSeq-ReqAndFbit</td>
<td>F = 0</td>
<td>Process-ReqSeq</td>
<td>WAIT_ACK</td>
</tr>
<tr>
<td>Recv Fbit</td>
<td>F = 1</td>
<td>Stop-MonitorTimer</td>
<td>XMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If UnackedFrames &gt; 0 then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-RetransTimer</td>
<td></td>
</tr>
<tr>
<td>Recv Fbit</td>
<td>F = 0</td>
<td></td>
<td>WAIT_ACK</td>
</tr>
<tr>
<td>MonitorTimer-Expires</td>
<td>RetryCount &lt; MaxTransmit</td>
<td>RetryCount := RetryCount+1</td>
<td>WAIT_ACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RRorRNR(P=1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-MonitorTimer</td>
<td></td>
</tr>
<tr>
<td>MonitorTimer-Expires</td>
<td>RetryCount ≥ MaxTransmit</td>
<td>Close Channel</td>
<td></td>
</tr>
</tbody>
</table>

*Table 8.4: WAIT_ACK State Table*

### 8.6.5.9 WAIT_F State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-Request</td>
<td>RemoteBusy = FALSE and RemWindow-Not-Full</td>
<td>Send-Data</td>
<td>WAIT_F</td>
</tr>
<tr>
<td>Data-Request</td>
<td>RemoteBusy = TRUE or RemWindow-Full</td>
<td>Pend-Data</td>
<td>WAIT_F</td>
</tr>
<tr>
<td>Recv ReqSeq-ReqAndFbit</td>
<td>F = 1</td>
<td>Process-ReqSeq</td>
<td>XMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop-MonitorTimer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If UnackedFrames &gt; 0 then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-RetransTimer</td>
<td></td>
</tr>
<tr>
<td>Recv ReqSeq-ReqAndFbit</td>
<td>F = 0</td>
<td>Process-ReqSeq</td>
<td>WAIT_F</td>
</tr>
<tr>
<td>Recv Fbit</td>
<td>F = 1</td>
<td>Stop-MonitorTimer</td>
<td>XMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If UnackedFrames &gt; 0 then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-RetransTimer</td>
<td></td>
</tr>
<tr>
<td>Recv Fbit</td>
<td>F = 0</td>
<td></td>
<td>WAIT_F</td>
</tr>
</tbody>
</table>

*Table 8.5: WAIT_F State Table*
### Table 8.5: WAIT_F State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonitorTimer-Expires</td>
<td>RetryCount &lt; MaxTransmit</td>
<td>RetryCount := RetryCount+1</td>
<td>WAIT_F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RR(P=1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start-MonitorTimer</td>
<td></td>
</tr>
<tr>
<td>MonitorTimer-Expires</td>
<td>RetryCount ≥ MaxTransmit</td>
<td>Close Channel</td>
<td></td>
</tr>
</tbody>
</table>
### Table 8.6: RECV State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recv I-frame (F=0)</strong></td>
<td>With-Expected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and LocalBusy = FALSE</td>
<td>Increment-ExpectedTxSeq PassToTx Data-Indication Send-Ack(F=0) If UnackedFrames = 0 then Stop-RetransTimer</td>
<td>RECV</td>
</tr>
<tr>
<td><strong>Recv I-frame (F=1)</strong></td>
<td>With-Expected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and LocalBusy = FALSE</td>
<td>Increment-ExpectedTxSeq PassToTx Data-Indication If RejActioned = FALSE then Retransmit-I-frames Send-Pending-I-frames else RejActioned := FALSE Send-Ack(F=0) If UnackedFrames = 0 then Stop-RetransTimer</td>
<td>RECV</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-duplicate-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and LocalBusy = FALSE</td>
<td>PassToTx</td>
<td>RECV</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-unexpected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and LocalBusy = FALSE</td>
<td>PassToTx SendREJ PassToTx InitSrei SaveframeSrej SendSREJ</td>
<td>REJ_SENT SREJ_SENT</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-Expected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and LocalBusy = TRUE</td>
<td>PassToTx StoreOrIgnore</td>
<td>RECV</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>and With-Valid-ReqSeq and Not-With_Expected_TxSeq and With-Valid-F-bit and LocalBusy = TRUE</td>
<td>PassToTx</td>
<td>RECV</td>
</tr>
<tr>
<td><strong>Recv RNR (P=0)</strong></td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := TRUE PassToTx Stop-RetransTimer</td>
<td>RECV</td>
</tr>
</tbody>
</table>
### Table 8.6: RECV_State table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
</table>
| Recv RNR (P=1) | With-Valid-ReqSeq and With-Valid-F-bit | RemoteBusy := TRUE  
PassToTx  
Stop-RetransTimer  
Send RRorRNR (F=1) | RECV        |
| Recv RR(P=0)(F=0) | With-Valid-ReqSeq and With-Valid-F-bit | PassToTx  
If RemoteBusy = TRUE then  
Retransmit-I-frames  
RemoteBusy :=FALSE  
Send-Pending-I-Frames  
If UnackedFrames = 0 then  
Stop-RetransTimer | RECV        |
| Recv RR(F=1) | With-Valid-ReqSeq and With-Valid-F-bit | RemoteBusy := FALSE  
PassToTx  
If RejActioned = FALSE then  
Retransmit-I-frames  
Send-Pending-I-frames  
else  
RejActioned := FALSE | RECV        |
| Recv RR(P=1) | With-Valid-ReqSeq and With-Valid-F-bit | PassToTx  
Send IorRRorRNR(F=1)  
RemoteBusy := FALSE | RECV        |
| Recv REJ (F=0) | With-Valid-ReqSeq  
Retrans and RetryIframes[i] < MaxTransmit  
and With-Valid-F-bit | RemoteBusy := FALSE  
PassToTx  
Retransmit-I-frames  
Send-Pending-I-frames  
If PbitOutstanding then  
RejActioned := TRUE | RECV        |
| Recv REJ (F=1) | With-Valid-ReqSeq  
Retrans and RetryIframes[i] < MaxTransmit  
and With-Valid-F-bit | RemoteBusy := FALSE  
PassToTx  
If RejActioned = FALSE then  
Retransmit-I-frames  
Send-Pending-I-frames  
else  
RejActioned := TRUE | RECV        |
<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
</table>
| Recv SREJ (P=0) (F=0) | With-Valid-ReqSeq-Retrans and RetryIframes[i] < Max-Transmit and With-Valid-F-bit | RemoteBusy := FALSE
PassToTxFbit
Retransmit-Requested-I-frame  
If PbitOutstanding then  
RejActioned := TRUE
SrejSaveReqSeq := ReqSeq | RECV        |
| Recv SREJ (P=0) (F=1) | With-Valid-ReqSeq-Retrans and RetryIframes[i] < Max-Transmit and With-Valid-F-bit | RemoteBusy := FALSE
PassToTxFbit  
If SrejActioned := TRUE and SrejSaveReqSeq := ReqSeq  
then  
SrejActions := FALSE  
else  
Retransmit-Requested-I-frame | RECV        |
| Recv SREJ(P=1)        | With-Valid-ReqSeq-Retrans and RetryIframes[i] < Max-Transmit and With-Valid-F-bit | RemoteBusy := FALSE
PassToTx
Retransmit-Requested-I-frame  
Send-Pending-I-frames  
If PbitOutstanding then  
SrejActioned = TRUE  
SrejSaveReqSeq := ReqSeq | RECV        |
| Recv REJ              | With-Valid-ReqSeq-Retrans and RetryIframes[i] < Max-Transmit               | Close Channel | RECV        |
| RECV SREJ             | With-Valid-ReqSeq-Retrans and RetryIframes[i] < Max-Transmit               | Close Channel | RECV        |
| Recv I-frame          | (With-Invalid-TxSeq and TxWindow > 32) or With-Invalid-ReqSeq               | Close Channel | RECV        |
| Recv I-frame          | With-Invalid-TxSeq and TxWindow ≤ 32                                       | Close Channel | RECV        |
| Recv RRorRNR          | With-Invalid-ReqSeq                                                         | Ignore       | RECV        |
| Recv REJorSREJ        | With-Invalid-ReqSeq-Retrans                                                | Close Channel | RECV        |

Table 8.6: RECV_State table
### Table 8.6: RECV_State table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recv frame</td>
<td></td>
<td>Ignore</td>
<td>RECV</td>
</tr>
</tbody>
</table>


### 8.6.5.11 REJ SENT State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recv I-frame (F=0)</td>
<td>With-Expected-TxSeq and With-Valid-ReqSeq and LocalBusy = FALSE</td>
<td>Increment-ExpectedTxSeq&lt;br&gt;PassToTx&lt;br&gt;Data-Indication&lt;br&gt;Send-Ack(F=0)&lt;br&gt;Stop-RetransTimer&lt;br&gt;If UnackedFrames = 0 then&lt;br&gt;Stop-RetransTimer</td>
<td>RECV</td>
</tr>
<tr>
<td>Recv I-frame (F=1)</td>
<td>With-unexpected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>Increment-ExpectedTxSeq&lt;br&gt;PassToTx&lt;br&gt;Data-Indication&lt;br&gt;Retransmit I-frames&lt;br&gt;Send-Pending-I-frames&lt;br&gt;RejActioned := FALSE&lt;br&gt;Send-Ack (F=0)&lt;br&gt;Stop-RetransTimer&lt;br&gt;If Unacked Frames = 0 then&lt;br&gt;Stop-RetransTimer</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td>Recv I-frame</td>
<td>With-unexpected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>PassToTx</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td>Recv RR (F=1)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := FALSE&lt;br&gt;PassToTx&lt;br&gt;Retransmit-I-frames&lt;br&gt;Send-Pending-I-frames&lt;br&gt;RejActioned := FALSE</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td>Recv RR(P=0)(F=0)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>PassToTx&lt;br&gt;RemoteBusy = True&lt;br&gt;Retransmit-I-frames&lt;br&gt;RemoteBusy := FALSE&lt;br&gt;Send-Ack(F=0)&lt;br&gt;Stop-RetransTimer&lt;br&gt;If UnackedFrames = 0 then&lt;br&gt;Stop-RetransTimer</td>
<td>REJ_SENT</td>
</tr>
</tbody>
</table>

Table 8.7: REJ_SENT State table
### Table 8.7: REJ_SENT State table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recv RR(P=1)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>PassToTx</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FramesSent := 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If RemoteBusy = True then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retransmit-I-frames</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RemoteBusy := FALSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If FramesSent = 0 then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RR(F=1)</td>
<td></td>
</tr>
<tr>
<td>Recv RNR(P=1)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := TRUE</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PassToTx</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RR(F=1)</td>
<td></td>
</tr>
<tr>
<td>Recv RNR(P=0)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := TRUE</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PassToTx</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send RR(F=0)</td>
<td></td>
</tr>
<tr>
<td>Recv REJ (F=0)</td>
<td>With-Valid-ReqSeq - Retrans and</td>
<td>RemoteBusy := FALSE</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td></td>
<td>RetryIframes[i] &lt; MaxTransmit and</td>
<td>PassToTx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>With-Valid-F-bit</td>
<td>Retransmit-I-frames</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send-Pending-I-frames</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If PbitOutstanding then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RejActioned := TRUE</td>
<td></td>
</tr>
<tr>
<td>Recv REJ (F=1)</td>
<td>With-Valid-ReqSeq - Retrans and</td>
<td>RemoteBusy := FALSE</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td></td>
<td>RetryIframes[i] &lt; MaxTransmit and</td>
<td>PassToTx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>With-Valid-F-bit</td>
<td>If RejActioned = FALSE then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retransmit-I-frames</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send-Pending-I-frames</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>else</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RejActioned := FALSE</td>
<td></td>
</tr>
<tr>
<td>Recv SREJ (P=0)</td>
<td>With-Valid-ReqSeq-Retrans and</td>
<td>RemoteBusy := FALSE</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td>(F=1)</td>
<td>RetryIframes[i] &lt; MaxTransmit and</td>
<td>PassToTx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>With-Valid-F-bit</td>
<td>If SrejActioned = TRUE and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SrejSaveReqSeq = ReqSeq</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>then</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SrejActioned := FALSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>else</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retransmit-Requested-I-frame</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Condition</td>
<td>Action</td>
<td>Next State</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| Recv SREJ (P=0) (F=0)  | With-Valid-ReqSeq-Retrans and RetryIframes[i] < MaxTransmit and With-Valid-F-bit | RemoteBusy := FALSE
PassToTxFbit
Retransmit-Requested-I-frames
If PbitOutstanding then
SrejActioned := TRUE
SrejSaveReqSeq := ReqSeq
RemoteBusy := FALSE
PassToTxFbit
If SrejActioned = TRUE and
SrejSaveReqSeq = ReqSeq
then
SrejActioned := FALSE
else
Send-Pending-I-frames | REJ_SENT                                                                   |
| Recv SREJ (P=0) (F=1)  | With-Valid-ReqSeq-Retrans and RetryIframes[i] < MaxTransmit and With-Valid-F-bit | RemoteBusy := FALSE
PassToTxFbit
If SrejActioned = TRUE and
SrejSaveReqSeq = ReqSeq
then
SrejActioned := FALSE
else
Send-Pending-I-frames | REJ_SENT                                                                   |
| Recv SREJ(P=1)         | With-Valid-ReqSeq-Retrans and RetryIframes[i] < MaxTransmit and With-Valid-F-bit | RemoteBusy := FALSE
PassToTx
Retransmit-Requested-I-frame
Send-Pending-I-frames
If PbitOutstanding then
SrejActioned := TRUE
SrejSaveReqSeq := ReqSeq
RemoteBusy := FALSE
PassToTxFbit
If SrejActioned = TRUE and
SrejSaveReqSeq = ReqSeq
then
SrejActioned := FALSE
else
Send-Pending-I-frames | REJ_SENT                                                                   |
| Recv REJ               | With-Valid-ReqSeq-Retrans and RetryIframes[i] < MaxTransmit               | Close Channel                                                           |              |
| Recv REJ (P=0)         | With-Valid-ReqSeq-Retrans and RetryIframes[i] ≥ MaxTransmit               | Close Channel                                                           | RECV         |
| RECV SREJ(P=1)         | With-Valid-ReqSeq-Retrans and RetryIframes[i] ≥ MaxTransmit               | Close Channel                                                           |              |
| Recv I-frame           | (With-Invalid-TxSeq and TxWindow > 32) or With-Invalid-ReqSeq              | Close Channel                                                           |              |
| Recv RRorRNR           | With-Invalid-ReqSeq                                                       | Close Channel                                                           |              |
| RecvRE,JorSREJ         | With-Invalid-ReqSeq-Retrans                                              | Close Channel                                                           |              |

Table 8.7: REJ_SENT State table
<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recv I-frame</td>
<td>With-Invalid-TxSeq and TxWindow ≤ 32</td>
<td>Close Channel</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td>Recv frame</td>
<td>Ignore</td>
<td>Ignore</td>
<td>RECV</td>
</tr>
</tbody>
</table>

Table 8.7: REJ_SENT State table
### 8.6.5.12 SREJ_SENT State Table

<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-Expected-TxSeq-Srej and With-Valid-ReqSeq and With-Valid-F-bit and SendRej = FALSE and SrejList = 1</td>
<td>SaveIframeSrej PopSrejList(TxSeq) PassToTx Data-IndicationSrej BufferSeq := BufferSeqSrej Send-Ack(F=0) If UnackedFrames = 0 then Stop-RetransTimer</td>
<td>RECV</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-Expected-TxSeq-Srej and With-Valid-ReqSeq and With-Valid-F-bit and SendRej = TRUE and SrejList = 1</td>
<td>SaveIframeSrej PopSrejList(TxSeq) PassToTx Data-IndicationSrej BufferSeq := BufferSeqSrej Send REJ</td>
<td>REJ_SENT</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-Expected-TxSeq-Srej and With-Valid-ReqSeq and SrejList &gt; 1</td>
<td>SaveIframe Increment-ExpectedTxSeq PassToTx</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-Unexpected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and SendRej = FALSE</td>
<td>SaveIframeSrej RemoveSrejList(TxSeq) PassToTx Send SREJ</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td><strong>Recv I-frame</strong></td>
<td>With-Unexpected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit and SendRej = TRUE</td>
<td>PassToTx SendRej := TRUE</td>
<td>SREJ_SENT</td>
</tr>
</tbody>
</table>

Table 8.8: SREJ_SENT State Table
<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recv I-frame</td>
<td>With-Unexpected-TxSeq and With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>SaveIframeSrej PassToTx Send SREJ(SrejList)</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td>Recv I-frame</td>
<td>With-duplicate-TxSeq-Sreq and With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>PassToTx</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td>Recv RR(F=1)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := FALSE PassToTx</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If RejActioned = FALSE then Retransmit-I-frames Send-Pending-I-frames else RejActioned := FALSE</td>
<td></td>
</tr>
<tr>
<td>Recv RR(P=1)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>PassToTx FramesSent :=0</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If RemoteBusy = TRUE then Retransmit-I-frames RemoteBusy := FALSE If FramesSent = 0 then Send SREJ(SrejList-tail)(F=1)</td>
<td></td>
</tr>
<tr>
<td>Recv RR(P=0)(F=0)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>PassToTx FramesSent :=0</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If RemoteBusy = TRUE then Retransmit-I-frames RemoteBusy := FALSE Send-Ack(F=0) If UnackedFrames = 0 then Stop-RetransTimer</td>
<td></td>
</tr>
<tr>
<td>Recv RNR(P=1)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := TRUE PassToTx</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send SREJ(SrejList-tail)(F=1)</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.8: SREJ_SENT State Table
<table>
<thead>
<tr>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recv RNR(P=0)</td>
<td>With-Valid-ReqSeq and With-Valid-F-bit</td>
<td>RemoteBusy := TRUE PassToTx Send RR(F=0)</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td>Recv REJ(F=0)</td>
<td>With-Valid-ReqSeq-Retrans and RetryIframes[i] &lt; MaxTransmit and With-Valid-F-bit</td>
<td>RemoteBusy := FALSE PassToTx Retransmit-I-frames Send-Pending-I-frames If PbitOutstanding then RejActioned := TRUE</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td>Recv REJ(F=1)</td>
<td>With-Valid-ReqSeq-Retrans and RetryIframes[i] &lt; MaxTransmit and With-Valid-F-bit</td>
<td>RemoteBusy := FALSE PassToTx If RejActioned = FALSE then Retransmit-I-frames Send-Pending-I-frames else RejActioned := FALSE</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td>Recv SREJ(P=0)(F=0)</td>
<td>With-Valid-ReqSeq-Retrans and RetryIframes[i] &lt; MaxTransmit and With-Valid-F-bit</td>
<td>RemoteBusy := FALSE PassToTx if PbitOutstanding then SrejSaveReqSeq = ReqSeq RejActioned := TRUE else</td>
<td>SREJ_SENT</td>
</tr>
<tr>
<td>Recv SREJ(P=0)(F=1)</td>
<td>With-Valid-ReqSeq-Retrans and RetryIframes[i] &lt; MaxTransmit and With-Valid-F-bit</td>
<td>RemoteBusy := FALSE PassToTx if SrejActioned = TRUE and SrejSaveReqSeq = ReqSeq then SrejActioned := FALSE else Retransmit-Requested-I-frame</td>
<td>SREJ_SENT</td>
</tr>
</tbody>
</table>

Table 8.8: SREJ_SENT State Table
8.7 STREAMING MODE

When a link is configured to work in Streaming Mode, the frame format for outgoing data is the same as for Enhanced Retransmission mode but frames are not acknowledged. Therefore

- RR, REJ, RNR and SREJ frames shall not be used in Streaming Mode.
• The F-bit shall always be set to zero in the transmitter, and shall be ignored in the receiver.

• the MonitorTimer and RetransmissionTimer shall not be used in Streaming mode.

A channel configured to work in Streaming mode shall be configured with a finite value for the Flush Timeout on the transmitter.

8.7.1 Transmitting I-frames

When transmitting a new I-frame the control field parameter ReqSeq shall be set to 0, TxSeq shall be set to NextTXSeq and NextTXSeq shall be incremented by one.

8.7.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be made available to the reassembly function. ExpectedTxSeq shall be incremented by one.

Upon receipt of a valid I-frame with an out-of-sequence TxSeq (see Section 8.7.3.1 on page 132) all frames with a sequence number less than TxSeq shall be assumed lost and marked as missing. The missing I-frames are in the range from ExpectedTxSeq (the frame that the device was expecting to receive) up to and including TxSeq-1. ExpectedTxSeq shall be set to TxSeq +1. The received I-frame shall be made available for pulling by the reassembly function. The ReqSeq shall be ignored.

Note: It is possible for a complete window size of I-frames to be missing and thus, no missing I-frames are detected. This situation occurs when 63 I-frames in a row are missing.

If there is no buffer space for the received I-frame an existing I-frame (i.e. the oldest) shall be discarded (flushed) freeing up buffer space for the new I-frame. The discarded I-frame shall be marked as missing.

8.7.3 Exception Conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in Streaming mode are defined in this section.

8.7.3.1 TxSeq Sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.
The out-of-sequence I-frame is identified by a TxSeq that is greater than ExpectedTxSeq (TxSeq > ExpectedTxSeq). The ReqSeq shall be ignored. The missing I-frame(s) are considered lost and ExpectedTXSeq is set equal to TxSeq+1 as specified in Section 8.7.2 on page 132. The missing I-frame(s) are reported as lost to the SDU reassembly function.
9 LIST OF FIGURES

Figure 1.1: L2CAP within protocol layers ..................................................... 11
Figure 1.2: L2CAP architectural blocks ....................................................... 12
Figure 2.1: Channels between devices ........................................................ 20
Figure 2.2: L2CAP transaction model ......................................................... 21
Figure 3.1: PDU format in Basic L2CAP mode on connection-oriented channels (field sizes in bits) ...................................................... 23
Figure 3.2: L2CAP PDU format in Basic L2CAP mode on Connectionless channel ................................................................................. 24
Figure 3.3: L2CAP PDU formats in Flow Control and Retransmission Modes ................................................................. 25
Figure 3.4: The LFSR circuit generating the FCS. ........................................ 31
Figure 3.5: Initial state of the FCS generating circuit. ................................. 31
Figure 4.1: L2CAP PDU format on the signaling channel .......................... 34
Figure 4.2: Command format ................................................................. 34
Figure 4.3: Command Reject packet .......................................................... 36
Figure 4.4: Connection Request Packet ..................................................... 37
Figure 4.5: Connection Response Packet .................................................. 39
Figure 4.6: Configuration Request Packet ................................................ 41
Figure 4.7: Configuration Request Flags field format ................................. 41
Figure 4.8: Configuration Response Packet ............................................ 43
Figure 4.9: Configuration Response Flags field format ............................. 43
Figure 4.10: Disconnection Request Packet ............................................. 45
Figure 4.11: Disconnection Response Packet ........................................... 46
Figure 4.12: Echo Request Packet ........................................................... 46
Figure 4.13: Echo Response Packet .......................................................... 47
Figure 4.14: Information Request Packet ............................................... 47
Figure 4.15: Information Response Packet ............................................. 48
Figure 5.1: Configuration option format .................................................. 50
Figure 5.2: MTU Option Format ............................................................. 51
Figure 5.3: Flush Timeout option format. .................................................. 52
Figure 5.4: Quality of Service (QoS) option format containing Flow Specification. ................................................................. 54
Figure 5.5: Retransmission and Flow Control option format. ............... 57
Figure 5.6: FCS option format. ............................................................... 63
Figure 6.1: States and transitions. ............................................................ 74
Figure 6.2: Configuration states and transitions. ..................................... 75
Figure 7.1: L2CAP fragmentation. ........................................................... 78
Figure 7.2: Example of fragmentation processes in a device with HCI. ........ 79
Figure 7.3: Segmentation and fragmentation of an SDU. .......................... 80
Figure 7.4: Example of segmentation and fragment processes in a device with HCI ................................................................. 81
List of Figures

Figure 8.1: Example of the transmitter side .................................................88
Figure 8.2: Example of the receiver side .....................................................90
Figure 8.3: Overview of the receiver side when operating in flow control mode .................................................................97
Figure 8.4: Receiver state machine ...........................................................105
10 LIST OF TABLES

Table 1.1: Terminology .......................................................................................... 16
Table 2.1: CID name space .................................................................................. 19
Table 2.2: Types of Channel Identifiers ............................................................... 20
Table 3.1: Standard Control Field formats ....................................................... 26
Table 3.2: Enhanced Control Field formats ....................................................... 27
Table 3.3: SAR control element format ............................................................... 28
Table 3.4: S control element format: type of S-frame ....................................... 28
Table 4.1: Signaling Command Codes ................................................................. 35
Table 4.2: Reason Code Descriptions ................................................................. 36
Table 4.3: Reason Data values .......................................................................... 37
Table 4.4: Result values ..................................................................................... 39
Table 4.5: Status values ..................................................................................... 40
Table 4.6: Configuration Response Result codes .............................................. 44
Table 4.7: InfoType definitions ......................................................................... 47
Table 4.8: Information Response Result values ............................................... 48
Table 4.9: Information Response Data fields ................................................... 49
Table 4.10: Extended feature mask .................................................................... 49
Table 5.1: Service type definitions .................................................................... 54
Table 5.2: Mode definitions ............................................................................... 57
Table 5.3: FCS types .......................................................................................... 63
Table 6.1: CLOSED state event table ................................................................. 65
Table 6.2: WAIT_CONNECT_RSP state event table ......................................... 66
Table 6.3: WAIT_CONNECT state event table ................................................ 66
Table 6.4: CONFIG state/substates event table: success transitions within configuration process .......................................................... 67
Table 6.5: CONFIG state/substates event table: non-success transitions within configuration process ........................................................ 68
Table 6.6: CONFIG state/substates event table: events not related to configuration process .......................................................... 69
Table 6.7: OPEN state event table ................................................................. 70
Table 6.8: WAIT_DISCONNECT state event table ......................................... 70
Table 7.1: Parameters allowed in Request ....................................................... 77
Table 7.2: Parameters allowed in Response ..................................................... 77
Table 8.1: Summary of actions when the RetransmissionTimer is in use and R=0. ........................................................................................... 92
Table 8.2: Enhanced Retransmission mode uses the following variables and sequence numbers described in Section 8.3 on page 86106 106
Table 8.3: XMIT state table ........................................................................... 117
Table 8.4: WAIT_ACK State Table ............................................................... 118
Table 8.5: WAIT_F State Table .................................................................... 118
Table 8.6: RECV_State table ....................................................................... 120
Logical Link Control and Adaptation Protocol Specification

Table 8.7: REJ_SENT State table .............................................................124
Table 8.8: SREJ_SENT State Table ........................................................128
APPENDIX A: CONFIGURATION MSCs

The examples in this appendix describe a sample of the multiple possible configuration scenarios that might occur.

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

![Diagram of basic MTU exchange](image)

Figure I: Basic MTU exchange

Figure II on page 140 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 141 illustrates an unsuccessful configuration request. There are two problems described by this example. The first problem is that 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. Device A then resends the configuration options using two smaller L2CAP_ConfigReq messages.

The second problem is an attempt to configure a channel with an invalid CID. For example device B may not have an open connection on that CID (0x01234567 in this example case).
Figure III: Unsuccessful Configuration Request