IEEE Std 802.15.1™-2005
(Revision of
IEEE Std 802.15.1-2002)

IEEE Standards

802.15.1™

IEEE Standard for
Information technology—
Telecommunications and information exchange between systems—
Local and metropolitan area networks—
Specific requirements

Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE

3 Park Avenue, New York, NY 10016-5997, USA

14 June 2005

Print: SH95323
PDF: SS95323

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Apple EX1026
Apple Inc. v. Secure Comm. Techs., LLC
IPR2026-00446

zed licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Recognized as an
American National Standard (ANSI)

IEEE Std 802.15.1™-2005
(Revision of
IEEE Std 802.15.1-2002)

IEEE Standard for Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements

Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 31 May 2005

American National Standards Institute

Approved 14 February 2005

IEEE-SA Standards Board

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Abstract: Methods for communicating devices in a personal area network (PAN) are covered in this standard.
Keywords: Bluetooth™, communications protocol, ISM, personal area network, WPAN

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2005 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 14 June 2005. Printed in the United States of America.

IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

Bluetooth is a registered trademark of Bluetooth SIG, Inc.

Print: ISBN 0-7381-4707-9 SH95323
PDF: ISBN 0-7381-4708-7 SS95323

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

NOTE—Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Introduction

This introduction is not part of IEEE Std 802.15.1-2005, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements: Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs).

This standard defines services and protocol elements that permit the exchange of management information between stations associated in a personal area network (PAN).

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates and nondiscriminatory, reasonable terms and conditions to applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates, terms, and conditions of the license agreements offered by patent holders or patent applicants. Further information may be obtained from the IEEE Standards Department.

Participants

The following is a list of participants in the Wireless Personal Area Networks Working Group at the time this standard was approved.

Robert F. Heile, *Chair, IEEE 802.15*
Thomas M. Siep, *Chair, Task Group 1a*

Jon Adams
Roberto Aiello
Hiroshi Akagi
Masaaki Akahane
Richard Alfvin
James Allen
Anand Anandakumar
Jong Hoon Ann
Mikio Aoki
Michimasa Aramaki
Larry Arnett
Arun Arunachalam
Naiel Askar
Venkatl Bahl
Yasaman Bahreini
Daniel Bailey
Jay Bain
James Baker
Jaiganesh Balakrishnan
Paul Ballentine
John Barr
Anuj Batra
Dagnachew Birru
Kenneth Boehlke
Monique Bourgeois
Mark Bowles
Charles Brabenac

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Vern Brethour
Ronald Brown
Peter Cain
Ed Callaway
Pat Carson
Kisoo Chang
Soo-Young Chang
Jonathon Cheah
Chee Wei Chew
Francois Po Shin Chin
Aik Chindapol
Sangsung Choi
Yun Choi
Craig Conkling
Robert Charles Cragie
David Cypher
Anand Dabak
Kai Dombrowski
Michael Dydyk
Jason Ellis
Shahriar Emami
Dwayne Escola
Mark W. Fidler
Chris Fisher
Reed Fisher
Jeff Foerster
Etsumi Fujita
Taketo Fukui
David Furuno
Pierre Gandolfo
Atul Garg
Michael Genossar
Vafa Ghazi
Ian Gifford
James Gilb
Tim Godfrey
Sorin Goldenberg
Paul Gorday
Martin Gravenstein
Evan Green
Bernd Grohmann
Assaf Gurevitz
Jose Gutierrez
Dongwoon Hahn
Thomas Hamilton
Yasuo Harada
Drew Harrington
Allen Heberling
Robert Heile
Barry Herold
Karl Heubaum
Reed Hinkel
Michael Hoghooghi
Srinath Hosur
Robert Huang
Eran Igler
Katsumi Ishii
Phil Jamieson
Ho-In Jeon
Jeyhan Karaoguz
Masami Katagiri
Joy Kelly
Michael Kelly
Stuart J. Kerry
Ryoji Kido
In Hwan Kim
Kyoung-A Kim
Myoung Soo Kim
Yongsuk Kim
Young Hwan Kim
Kursat Kimyacioglu
Patrick Kinney
Guenter Kleindl
Toshiya Kobashi
Ryuji Kohno
Bruce P. Kraemer
Rajeev Krishnamoorthy
Do-Hoon Kwon
John V. Lampe
Jim Lansford
Torbjorn Larsson
Hyung Soo Lee
Myung Lee
Nag Lee
Simon Lee
Woo-Kyung Lee
David Leeper
Liang Li
Susan Lin
Hui-Ling Lou
Akira Maeki
Steven March
Frederick Martin
Noriaki Matsuno
John McCorkle
Michael McInnis
Michael McLaughlin
James McLean
Daniel Meacham
Jim Meyer
Leonard Miller
Akira Miura
Shaomin Mo
Andreas Molisch
Antonio Mondragon
Mark Moore
Anthony Morelli
Said Moridi
Steven Morton
Marco Naeve
Ken Naganuma
Yves-Paul Nakache
Chiu Ngo
Erwin R. Noble
Christopher O'Connor
John C. O'Conor
Knut Odman
Hiroyo Ogawa
Eric Ojard
John B. Pardee
Jonghun Park
Joon Goo Park
Dave Patton
Marcus Pendergrass
Xiaoming Peng
Robert D. Poor
Paul Popescu
Pekka Ranta
Yaron Rashi
Gregg Rasor
Charles Razzell
Ivan Reede
Glyn Roberts
Richard Roberts
Martin Rofheart
Chris Rogers
Gerald Rogerson
Philippe Rouzet
Chandos Rypinski
Shin Saito
Tomoki Saito
John Santhoff
John Sarallo
Yasufumi Sasaki
Mark Schrader
Tom Schuster
Erik Schylander
Michael Seals
Kevin Shelby
Stephen J. Shellhammer
Shusaku Shimada
Cheol-Ho Shin
Yuichi Shiraki
Etan Shirron
Gadi Shor
William Shvodian
Thomas Siep
Kazimierz Siwiak
V. Somayazulu
Carl Stevenson
Rene Struik
Hiroto Sugahara
Robert A. Sutton
MItsuhiro Suzuki
Katsumi Takaoka
Kenichi Takizawa
Teik-Kheong Tan
Mike Tanahashi
James Taylor
Jerome Tjia
Kiyohito Tokuda
Jean Tsao
Stephen Turner
Hans van Leeuwen
Bhupender Virk
Thierry Walrant
Jerry Wang
Jing Wang
Fujio Watanabe
Katsumi Watanabe
Matthew Welborn
Richard Wilson
Gerald Wineinger
Stephen Wood
Edward G. Woodrow
David Yaish
Hirohisa Yamaguchi
Wonyong Yoon
Amos Young
Song-Lin Young
Serdar Yurdakul
Honggang Zhang
James Zyren

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The following members of the balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Jon Adams
Jaemin Ahn
Roberto Aiello
Hiroshi Akagi
Richard Alfvin
James Allen
Richard Allen
Anand Anandakumar
Jong Hoon Ann
Mikio Aoki
Larry Arnett
Naiel Askar
Venkatb Bahl
Yasaman Bahreini
Jay Bain
James Baker
Jaiganesh Balakrishnan
Kannan Balakrishnan
Paul Ballentine
John Barr
Anuj Batra
Phil Beecher
Dagnachew Birru
Kenneth Boehlke
Herve Bonneville
John Boot
Bruce Bosco
Monique Bourgeois
Mark Bowles
Charles Brabenac
Jennifer Bray
David Brenner
Vern Brethour
Ronald Brown
Peter Cain
Ed Callaway
Pat Carson
Kisoo Chang
Soo-Young Chang
Jonathon Cheah
CheeWei Chew
Francois Chin
Yu-Chang Chiu
Sarm Cho
Sangsung Choi
Yun Choi
Chia-Chin Chong
Manoj Choudhary
Craig Conkling
Celestino Corral
Robert Cragie
David Cypher
Anand Dabak
Scott Davis
Joe Decuir
Javier Del Prado Pavon
Kai Dombrowski
Stefan Drude
Eryk Dutkiewicz
Michael Dydyk
Jason Ellis
Shahriar Emami
Yossi Erlich
Dwayne Escola
Mark Fidler
Chris Fisher
Reed Fisher
Kristoffer Fleming
Etsumi Fujita
Taketo Fukui
David Furuno
Ricardo Gandia Sanchez
Pierre Gandolfo
Michael Genossar
Vafa Ghazi
Ian Gifford
James Gilb
Tim Godfrey
Sung-Wook Goh
Sorin Goldenberg
Paul Gorday
Martin Gravenstein
Evan Green
Bernd Grohmann
Assaf Gurevitz
Dongwoon Hahn
Julian Hall
Thomas Hamilton
Yasuo Harada
Drew Harrington
Jeff Harris
Amer Hassan
Vann Hasty
Allen Heberling
Robert Heile
Barry Herold
Karl Heubaum
Jin-Meng Ho
Michael Hoghooghi
Srinath Hosur
Patrick Houghton
Chi-Hao Huang
Robert Huang
Xiaojing Huang
Eran Igler
Tetsushi Ikegami
Yeong Min Jang
Bruno Jechoux
Ho-In Jeon
Tzyy Hong Jiang (Chiang)
Peter Johansson
Jeyhan Karaoguz
Masami Katagiri
Joy Kelly
Michael Kelly
Stuart Kerry
Ryoji Kido
Haksun Kim
Inhwan Kim
Jae Young Kim
Myoung Kim
Yongsuk Kim
Young Hwan Kim
Youngsoo Kim
Kursat Kimyacioglu
Matthias Kindler
Patrick Kinney
Guenter Kleindl
Toshiya Kobashi
Noam Kogos
Ryuji Kohno
Rajeev Krishnamoorthy
Haim Kupershmidt
Yuzo Kuramochi
Do-Hoon Kwon
Taekyoung Kwon
Kang KyuMin
John Lampe
Jim Lansford
Torbjorn Larsson
David Leach
Dongjun Lee
Hyung Soo Lee
Kyung-Kuk Lee
Myung Lee
Nag Lee
Simon Lee
Woo-Kyung Lee
David Leeper
Fabrice Legrand
Israel Leibovich
Henry Li
Huan-Bang Li
Liang Li
Jie Liang
Susan Lin
Yong Liu
Hui-Ling Lou
Darryn Lowe
Steve Ma
Tadahiko Maeda
Akira Maeki
Frederick Martin
Abbie Mathew
Masafumi Matsumura
John McCorkle
Michael McInnis
Michael McLaughlin
James McLean
Daniel Meacham
Jim Meyer
Leonard Miller
Akira Miura
Hitoshi Miyasaka
Shaomin Mo
Andreas Molisch
Antonio Mondragon
Mark Moore
Anthony Morelli
Steven Morton
Marco Naeve
Ken Naganuma
Hiroyuki Nagasaka

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Yves-Paul Nakache
Chiu Ngo
Erwin Noble
Mizukoshi Nobuyuki
Masaki Noda
Richard Noens
Christopher O'Connor
John (Jay) O'Conor
Knut Odman
Hiroyo Ogawa
Eric Ojard
Philip Orlik
Eiichiro Otobe
John Pardee
Bonghyuk Park
Jonghun Park
Joon Goo Park
Young Jin Park
Vijay Patel
Dave Patton
Miguel Pellon
Xiaoming Peng
Robert Poor
Paul Popescu
Clinton Powell
Raad Raad
Ajay Rajkumar
Pekka Ranta
Yaron Rashi
Gregg Rasor
Charles Razzell
Ivan Reede
Mark Rich
Yuko Rikuta
Benno Ritter
Terry Robar
Glyn Roberts
Richard Roberts
Martin Rofheart
Christopher Rogers
Gerald Rogerson
Jaeho Roh
Philippe Rouzet
Chandos Rypinski
Zafer Sahinoglu
Tomoki Saito
John Santhoff
John Sarallo
Yasufumi Sasaki
Sidney Schrum
Tom Schuster
Erik Schylander
Michael Seals
Huai-Rong Shao
Kevin Shelby
Stephen Shellhammer
Chih-Chung Shi
Shusaku Shimada
Cheol-Ho Shin
Yuichi Shiraki
Etan Shirron
Matthew Shoemake
Gadi Shor
William Shvodian
Thomas Siep
Michael Sim
Kazimierz Siwiak
Yoram Solomon
V. Somayazulu
Amjad Soomro
Carl Stevenson
Marinus Struik
Hiroto Sugahara
Robert Sutton
MItsuhiro Suzuki
Kazuaki Takahashi
Kenichi Takizawa
Teik-Kheong Tan
Mike Tanahashi
James Taylor
John Terry
Jerome Tjia
Kiyohito Tokuda
Jean Tsao
Stephen Turner
Oltac Unsal
Hans Van Leeuwen
Bhupender Virk
Timothy Wakeley
Thierry Walrant
Vivek Wandile
Jerry Wang
Jing Wang
Yunbiao Wang
Fujio Watanabe
Chris Weber
Matthew Welborn
Richard Wilson
Gerald Wineinger
Andreas Wolf
Timothy Wong
Stephen Wood
Patrick Worfolk
Xiaodong Wu
Yu-Ming Wu
David Yaish
Hirohisa Yamaguchi
Kamya Yekeh Yazdandoost
Wonyong Yoon
Yutaka Yoshida
Song-Lin Young
Hon Yung
Serdar Yurdakul
Honggang Zhang
Frank Xiaojun Zheng
Chunhui Zhu
James Zyren

When the IEEE-SA Standards Board approved this standard on 14 February 2005, it had the following membership:

Don Wright, *Chair*
Steve M. Mills, *Vice Chair*
Judith Gorman, *Secretary*

Chuck Adams
Stephen Berger
Mark D. Bowman
Joseph A. Bruder
Bob Davis
Roberto de Marca Boisson
Julian Forster*
Arnold M. Greenspan
Mark S. Halpin
Raymond Hapeman
Richard J. Holleman
Richard H. Hulett
Lowell G. Johnson
Joseph L. Koepfinger*
Hermann Koch
Thomas J. McGean
Daleep C. Mohla
Paul Nikolich
T. W. Olsen
Ronald C. Petersen
Gary S. Robinson
Frank Stone
Malcolm V. Thaden
Doug Topping
Joe D. Watson

*Member Emeritus

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan Cookson, *NIST Representative*

Don Messina
IEEE Standards Project Editor

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Contents

1. Overview ... 1

1.1 Scope ... 1
1.2 WPAN definition ... 1

2. Normative references ... 3

2.1 IEEE documents ... 3
2.2 ISO documents ... 3
2.3 ITU documents ... 3
2.4 Other documents ... 4

3. Definitions ... 5

4. Acronyms and abbreviations ... 11

4.1 Standard-based acronyms and abbreviations ... 11
4.2 Bluetooth specification names ... 14

5. General description ... 17

5.1 New features ... 17
5.2 Changes in wording ... 17
5.2.1 IEEE language update ... 17
5.2.2 Nomenclature changes ... 17
5.3 Structure changes ... 18
5.4 Deprecated features ... 18

6. Architecture ... 19

6.1 General description ... 19
6.2 Core system architecture ... 20
6.3 Core architectural blocks ... 22
6.3.1 Channel manager ... 22
6.3.2 L2CAP resource manager ... 22
6.3.3 Device manager ... 22
6.3.4 Link manager (LM) ... 22
6.3.5 BB resource manager ... 22
6.3.6 Link controller ... 23
6.3.7 Radio frequency (RF) ... 23
6.4 Data transport architecture ... 23
6.4.1 Core traffic bearers ... 24
6.4.2 Transport architecture entities ... 27
6.4.3 Generic packet structure ... 28
6.4.4 Physical channels ... 29
6.4.5 Physical links ... 32
6.4.6 Logical links and logical transports ... 33
6.5 L2CAP channels ... 39
6.6 Communication topology ... 40
6.6.1 Piconet topology ... 40
6.6.2 Operational procedures and modes ... 41

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

7. Physical layer (PHY) 45

7.1 Scope 45
7.1.1 Regional authorities 45
7.1.2 Frequency bands and channel arrangement 45
7.2 Transmitter characteristics 46
7.3 Modulation characteristics 47
7.3.1 Spurious emissions 47
7.4 Receiver characteristics 49
7.4.1 Actual sensitivity level 49
7.4.2 Interference performance 49
7.4.3 Out-of-band blocking 49
7.4.4 Intermodulation characteristics 50
7.4.5 Maximum usable level 50
7.4.6 Receiver signal strength indicator 50
7.4.7 Reference signal definition 50
7.5 Nominal test conditions 51
7.5.1 Nominal temperature 51
7.5.2 Nominal power source 51
7.6 Extreme test conditions 51
7.6.1 Extreme temperatures 51
7.6.2 Extreme power source voltages 51
7.7 Test condition parameters 52

8. Baseband (BB) 53

8.1 General description 53
8.1.1 Clock 53
8.1.2 Device addressing 54
8.1.3 Access codes 55
8.2 Physical channels 55
8.2.1 Physical channel definition 56
8.2.2 Basic piconet physical channel 56
8.2.3 Adapted piconet physical channel 61
8.2.4 Page scan physical channel 61
8.2.5 Inquiry scan physical channel 64
8.2.6 Hop selection 66
8.3 Physical links 75
8.3.1 Link supervision 75
8.4 Logical transports 76
8.4.1 General 76
8.4.2 Logical transport address (LT_ADDR) 76
8.4.3 Synchronous logical transports 77
8.4.4 Asynchronous logical transport 77
8.4.5 Transmit/receive routines 77
8.4.6 Active slave broadcast (ASB) transport 78
8.4.7 Parked slave broadcast (PSB) transport 78
8.5 Logical links 78
8.5.1 Link control (LC) logical link 79
8.5.2 ACL control (ACL-C) logical link 79
8.5.3 Asynchronous/Isochronous user (ACL-U) logical link 79
8.5.4 Stream logical link 79
8.5.5 Logical link priorities 80
8.6 Packets 80

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.1 General format 80
8.6.2 Bit ordering 80
8.6.3 Access code 80
8.6.4 Packet header 85
8.6.5 Packet types 86
8.6.6 Payload format 92
8.6.7 Packet summary 95
8.7 Bitstream processing 96
8.7.1 Error checking 97
8.7.2 Data whitening 99
8.7.3 Error correction 100
8.7.4 FEC code: rate 1/3 100
8.7.5 FEC code: rate 2/3 100
8.7.6 Automatic repeat request (ARQ) scheme 101
8.8 Link controller operation 108
8.8.1 Overview of states 108
8.8.2 STANDBY state 109
8.8.3 Connection establishment substates 109
8.8.4 Device discovery substates 115
8.8.5 Connection state 118
8.8.6 Active mode 119
8.8.7 SNIFF mode 129
8.8.8 HOLD mode 131
8.8.9 PARK state 131
8.9 Audio 137
8.9.1 Log PCM coder decoder (CODEC) 137
8.9.2 CVSD CODEC 137
8.9.3 Error handling 139
8.9.4 General audio requirements 139
8.10 General audio recommendations 140
8.10.1 Maximum sound pressure 140
8.10.2 Other telephony network requirements 140
8.10.3 Audio levels 140
8.10.4 Microphone path 141
8.10.5 Loudspeaker path 141
8.10.6 Voice interface 141
8.10.7 Frequency mask 141
8.11 Timers 143
8.11.1 inquiryTO 143
8.11.2 pageTO 143
8.11.3 pagerespTO 143
8.11.4 newconnectionTO 143
8.11.5 supervisionTO 143
8.12 Recommendations for AFH operation in PARK, HOLD, and SNIFF 143
8.12.1 Operation at the master 144
8.12.2 Operation in PARK state 144
8.12.3 AFH operation in SNIFF mode 145
8.12.4 AFH operation in HOLD mode 145

9. Link Manager Protocol (LMP) 147

9.1 General rules 147
9.1.1 Message transport 147
9.1.2 Synchronization 147

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.1.3 Packet format ... 148
9.1.4 Transactions ... 149
9.1.5 Error handling ... 150
9.1.6 Procedure rules ... 150
9.1.7 General response messages ... 151
9.1.8 LMP message constraints ... 152
9.2 Device features ... 152
9.2.1 Feature definitions ... 152
9.2.2 Features mask definition ... 154
9.2.3 LM interoperability policy ... 156
9.3 Procedure rules ... 156
9.3.1 Connection control ... 156
9.3.2 Security ... 168
9.3.3 Informational requests ... 177
9.3.4 Role switch ... 181
9.3.5 Modes of operation ... 183
9.3.6 Logical transports ... 192
9.3.7 Test mode ... 198
9.4 Summary ... 199
9.4.1 PDU summary ... 199
9.4.2 Parameter definitions ... 209
9.4.3 Default values ... 214

10. Error codes ... 215

10.1 HCI command errors ... 215
10.2 List of error codes ... 215
10.3 Error code descriptions ... 217
10.3.1 Unknown HCI command (0x01) ... 217
10.3.2 Unknown connection identifier (0x02) ... 217
10.3.3 Hardware failure (0x03) ... 217
10.3.4 Page timeout (0x04) ... 217
10.3.5 Authentication failure (0x05) ... 217
10.3.6 PIN missing (0x06) ... 217
10.3.7 Memory capacity exceeded (0x07) ... 218
10.3.8 Connection timeout (0x08) ... 218
10.3.9 Connection limit exceeded (0x09) ... 218
10.3.10 Synchronous connection limit to a device exceeded (0x0A) ... 218
10.3.11 ACL connection already exists (0x0B) ... 218
10.3.12 Command disallowed (0x0C) ... 218
10.3.13 Connection rejected due to limited resources (0x0D) ... 218
10.3.14 Connection rejected due to security reasons (0x0E) ... 218
10.3.15 Connection rejected due to unacceptable BD_ADDR (0x0F) ... 218
10.3.16 Connection accept timeout exceeded (0x10) ... 218
10.3.17 Unsupported feature or parameter value (0x11) ... 219
10.3.18 Invalid HCI command parameters (0x12) ... 219
10.3.19 Remote user terminated connection (0x13) ... 219
10.3.20 Remote device terminated connection due to low resources (0x14) ... 219
10.3.21 Remote device terminated connection due to power off (0x15) ... 219
10.3.22 Connection terminated by local host (0x16) ... 219
10.3.23 Repeated attempts (0x17) ... 219
10.3.24 Pairing not allowed (0x18) ... 219
10.3.25 Unknown LMP PDU (0x19) ... 219
10.3.26 Unsupported remote feature (0x1A) ... 220

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10.3.27 SCO offset rejected (0x1B) ... 220
10.3.28 SCO interval rejected (0x1C) ... 220
10.3.29 SCO air mode rejected (0x1D) ... 220
10.3.30 Invalid LMP parameters (0x1E) ... 220
10.3.31 Unspecified error (0x1F) ... 220
10.3.32 Unsupported LMP parameter value (0x20) ... 220
10.3.33 Role change not allowed (0x21) ... 220
10.3.34 LMP response timeout (0x22) ... 220
10.3.35 LMP error transaction collision (0x23) ... 220
10.3.36 LMP PDU not allowed (0x24) ... 221
10.3.37 Encryption mode not acceptable (0x25) ... 221
10.3.38 Link key cannot be changed (0x26) ... 221
10.3.39 Requested QoS not supported (0x27) ... 221
10.3.40 Instant passed (0x28) ... 221
10.3.41 Pairing with unit key not supported (0x29) ... 221
10.3.42 Different transaction collision (0x2A) ... 221
10.3.43 QoS unacceptable parameter (0x2C) ... 221
10.3.44 QOS rejected (0x2D) ... 221
10.3.45 Channel classification not supported (0x2E) ... 221
10.3.46 Insufficient security (0x2F) ... 221
10.3.47 Parameter out of mandatory range (0x30) ... 222
10.3.48 Role switch pending (0x32) ... 222
10.3.49 Reserved slot violation (0x34) ... 222
10.3.50 Role switch failed (0x35) ... 222

11. Host controller interface (HCI) ... 223

11.1 Lower layers of the IEEE 802.15.1-2005 software stack ... 223
11.2 Overview of host controller transport ... 224
11.3 Overview of commands and events ... 224
11.3.1 Generic events ... 225
11.3.2 Device setup ... 225
11.3.3 Controller flow control ... 225
11.3.4 Controller information ... 225
11.3.5 Controller configuration ... 226
11.3.6 Device discovery ... 227
11.3.7 Connection setup ... 227
11.3.8 Remote information ... 229
11.3.9 Synchronous connections ... 229
11.3.10 Connection state ... 230
11.3.11 Piconet structure ... 231
11.3.12 QoS ... 231
11.3.13 Physical links ... 232
11.3.14 Host flow control ... 233
11.3.15 Link information ... 233
11.3.16 Authentication and encryption ... 234
11.3.17 Testing ... 235
11.3.18 Alphabetical list of commands and events ... 236
11.4 HCI flow control ... 241
11.4.1 Host-to-controller data flow control ... 241
11.4.2 Controller-to-host data flow control ... 241
11.4.3 Disconnection behavior ... 242
11.4.4 Command flow control ... 242
11.4.5 Command error handling ... 243

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.5 HCI data formats 243
11.5.1 Introduction 243
11.5.2 Data and parameter formats 243
11.5.3 Connection handles 244
11.5.4 Exchange of HCI-specific information 245
11.6 HCI configuration parameters 249
11.6.1 Scan_Enable 249
11.6.2 Inquiry_Scan_Interval 249
11.6.3 Inquiry_Scan_Window 250
11.6.4 Inquiry_Scan_Type 250
11.6.5 Inquiry_Mode 250
11.6.6 Page_Reply_Timeout 251
11.6.7 Connection_Accept_Timeout 251
11.6.8 Page_Scan_Interval 251
11.6.9 Page_Scan_Window 252
11.6.10 Page_Scan_Period_Mode 252
11.6.11 Page_Scan_Type 252
11.6.12 Voice_Setting 253
11.6.13 PIN_Type 254
11.6.14 Link_Key 254
11.6.15 Authentication_Enable 254
11.6.16 Encryption_Mode 255
11.6.17 Failed_Contact_Counter 255
11.6.18 HOLD_Mode_Activity 256
11.6.19 Link_Policy_Settings 256
11.6.20 Flush_Timeout 257
11.6.21 Number_of_Broadcast_Retransmissions 257
11.6.22 Link_Supervision_Timeout 257
11.6.23 Synchronous_Flow_Control_Enable 258
11.6.24 Local_Name 258
11.6.25 Class_of_Device 259
11.6.26 Supported_Commands 259
11.7 HCI commands and events 263
11.7.1 Link control commands 264
11.7.2 Link policy commands 293
11.7.3 Controller-BB commands 306
11.7.4 Informational parameters 354
11.7.5 Status parameters 359
11.7.6 Testing commands 365
11.7.7 Events 368
11.8 Deprecated commands, events, and configuration parameters 395
11.8.1 Page_Scan_Mode parameter 396
11.8.2 Read Page Scan Mode command 396
11.8.3 Write Page Scan Mode command 396
11.8.4 Read Country Code command 397
11.8.5 Add SCO Connection command 398
11.8.6 Page Scan Mode Change event 399

12. Message sequence charts (MSCs) 401

12.1 Overview 401
12.1.1 Notation 401
12.1.2 Flow of control 401
12.1.3 Sample MSC 402

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.2 Services without connection request ... 402
12.2.1 Remote name request ... 402
12.2.2 One-time inquiry ... 403
12.2.3 Periodic inquiry ... 405
12.3 ACL Connection establishment and detachment ... 406
12.3.1 Connection setup ... 407
12.4 Optional activities after ACL connection establishment ... 413
12.4.1 Authentication requested ... 413
12.4.2 Set connection encryption ... 414
12.4.3 Change connection link key ... 415
12.4.4 Master link key ... 416
12.4.5 Read remote supported features ... 418
12.4.6 Read remote extended features ... 418
12.4.7 Read clock offset ... 419
12.4.8 Read remote version information ... 419
12.4.9 QoS setup ... 420
12.4.10 Switch role ... 420
12.5 Synchronous connection establishment and detachment ... 421
12.5.1 Synchronous connection setup ... 421
12.6 SNIFF, HOLD, and PARK ... 426
12.6.1 SNIFF mode ... 426
12.6.2 HOLD mode ... 427
12.6.3 PARK state ... 429
12.7 Buffer management, flow control ... 432
12.8 Loopback mode ... 433
12.8.1 Local loopback mode ... 433
12.8.2 Remote loopback mode ... 435

13. Security ... 437

13.1 Security overview ... 437
13.2 Random number generation ... 438
13.3 Key management ... 438
13.3.1 Key types ... 438
13.3.2 Key generation and initialization ... 440
13.4 Encryption ... 444
13.4.1 Encryption key size negotiation ... 445
13.4.2 Encryption of broadcast messages ... 445
13.4.3 Encryption concept ... 446
13.4.4 Encryption algorithm ... 447
13.4.5 LFSR initialization ... 449
13.4.6 Key stream sequence ... 452
13.5 Authentication ... 452
13.5.1 Repeated attempts ... 453
13.6 The authentication and key-generating functions ... 454
13.6.1 The authentication function E1 ... 454
13.6.2 The functions Ar and A'r ... 455
13.6.3 E2-key generation function for authentication ... 457
13.6.4 E3-key generation function for encryption ... 459

14. Logical Link Control and Adaptation Protocol (L2CAP) ... 461

14.1 L2CAP features ... 461
14.1.1 Assumptions ... 463

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.1.2 Scope ... 464
14.1.3 Terminology ... 464
14.2 General operation ... 466
14.2.1 Channel identifiers (CIDs) ... 466
14.2.2 Operation between devices ... 467
14.2.3 Operation between layers ... 468
14.2.4 Modes of operation ... 468
14.3 Data packet format ... 469
14.3.1 Connection-oriented channel in basic L2CAP mode ... 469
14.3.2 Connectionless data channel in basic L2CAP mode ... 469
14.3.3 Connection-oriented channel in retransmission/flow control modes ... 470
14.4 Signalling packet formats ... 474
14.4.1 Command Reject packet (code 0x01) ... 475
14.4.2 Connection Request packets (code 0x02) ... 477
14.4.3 Connection Response packet (code 0x03) ... 478
14.4.4 Configuration Request packet (code 0x04) ... 479
14.4.5 Configuration Response packet (code 0x05) ... 480
14.4.6 Disconnection Request packet (code 0x06) ... 482
14.4.7 Disconnection Response packet (code 0x07) ... 483
14.4.8 Echo Request packet (code 0x08) ... 483
14.4.9 Echo Response packet (code 0x09) ... 484
14.4.10 Information Request packet (code 0x0a) ... 484
14.4.11 Information Response packet (code 0x0b) ... 485
14.4.12 Extended features mask ... 486
14.5 Configuration parameter options ... 486
14.5.1 MTU option ... 487
14.5.2 Flush timeout option ... 488
14.5.3 QoS option ... 489
14.5.4 Retransmission and flow control option ... 491
14.6 State machine ... 493
14.6.1 General rules for the state machine ... 493
14.6.2 Timers events ... 501
14.7 General procedures ... 502
14.7.1 Configuration process ... 503
14.7.2 Fragmentation and recombination ... 504
14.7.3 Encapsulation of SDUs ... 505
14.7.4 Delivery of erroneous L2CAP SDUS ... 507
14.7.5 Operation with flushing ... 507
14.7.6 Connectionless data channel ... 508
14.8 Procedures for flow control and retransmission ... 508
14.8.1 Information retrieval ... 508
14.8.2 Function of PDU types for flow control and retransmission ... 508
14.8.3 Variables and SEQNs ... 509
14.8.4 Retransmission mode ... 512
14.8.5 Flow control mode ... 516
14.9 Configuration MSCs ... 519

15. Service access point (SAP) interfaces and primitives ... 523

15.1 IEEE 802® interfaces ... 523
15.1.1 LLC sublayer service specifications (general) ... 525
15.2 LLC sublayer/MAC sublayer interface service specification ... 526
15.2.1 MA-UNITDATA request ... 526
15.2.2 MA-UNITDATA indication ... 527

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.2.3 MA-UNITDATA-STATUS indication 528
15.3 Bluetooth interfaces 529
15.3.1 MSC of layer interactions 530
15.3.2 Relationship of Bluetooth protocol entities to IEEE 802 constructs 530
15.3.3 Upper layer interface definitions 537
15.3.4 Service primitives 538

Annex A (informative) Bibliography 553

Annex B (informative) Generic access profile (GAP) 555

B.1 Scope 555
B.2 Symbols and conventions 555
B.2.1 Requirement status symbols 555
B.2.2 Signaling diagram conventions 556
B.2.3 Notation for timers and counters 557
B.3 Profile overview 557
B.3.1 Profile stack 557
B.3.2 Configurations and roles 557
B.3.3 User requirements and scenarios 558
B.3.4 Profile fundamentals 558
B.4 Modes 558
B.4.1 Discoverability modes 559
B.4.2 Connectability modes 561
B.4.3 Pairing modes 562
B.5 Security aspects 562
B.5.1 Authentication 563
B.5.2 Security modes 563
B.6 Idle mode procedures 566
B.6.1 General inquiry 566
B.6.2 Limited inquiry 567
B.6.3 Name discovery 568
B.6.4 Bonding 570
B.7 Establishment procedures 572
B.7.1 Link establishment 573
B.7.2 Channel establishment 575
B.7.3 Connection establishment 576
B.8 Timers and constants 577
B.9 Information flows of related procedures 578
B.9.1 LMP authentication 578
B.9.2 LMP pairing 578
B.9.3 Service discovery (SD) 579

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

IEEE Standard for
Information technology—

Telecommunications and information exchange between systems—

Local and metropolitan area networks—

Specific requirements

Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)

1. Overview

Wireless personal area networks (WPANs) are used to convey information over short distances among a private, intimate group of participant devices. Unlike a wireless local area network (WLAN), a connection made through a WPAN involves little or no infrastructure or direct connectivity to the world outside the link. This allows small, power-efficient, inexpensive solutions to be implemented for a wide range of devices.

1.1 Scope

This standard defines physical layer (PHY) and medium access control (MAC) specifications for wireless connectivity with fixed, portable, and moving devices within or entering a personal operating space (POS). A POS is the space about a person or object that typically extends up to 10 m in all directions and envelops the person whether stationary or in motion.

The original goal of the IEEE 802.15.1 Task Group was to achieve a level of interoperability that could allow the transfer of data between a WPAN device and an IEEE 802.11™ device. Although this proved infeasible, IEEE Std 802.15.1-2005 does have mechanisms defined to allow better coexistence with IEEE 802.11b™ class of devices.

Both this standard and the previous version are based upon technology originally developed by the Bluetooth™ Special Interest Group (SIG).

1.2 WPAN definition

The term *WPAN* in this standard refers specifically to a wireless personal area network as used in this standard.

Specifically, this standard describes the following:

— The functions and services required by an IEEE 802.15.1-2005 device to operate within ad hoc networks.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

- The following MAC procedures to support the asynchronous connectionless or connection-oriented (ACL) and synchronous connection-oriented (SCO) link delivery services:
  - The baseband (BB) layer, specifying the lower level operations at the bit and packet levels, e.g., forward error correction (FEC) operations, encryption, cyclic redundancy check (CRC) calculations, Automatic Repeat Request (ARQ) Protocol.
  - The link manager (LM) layer, specifying connection establishment and release, authentication, connection and release of SCO and ACL channels, traffic scheduling, link supervision, and power management tasks.
  - The Logical Link Control and Adaptation Protocol (L2CAP) layer, forming an interface to standard data transport protocols. It handles the multiplexing of higher layer protocols and the segmentation and reassembly (SAR) of large packets. The data stream crosses the LM layer, where packet scheduling on the ACL channel takes place. The audio stream is directly mapped on an SCO channel and bypasses the LM layer. The LM layer, though, is involved in the establishment of the SCO link. Between the LM layer and the application, control messages are exchanged in order to configure the IEEE 802.15.1-2005 transceiver for the considered application.
- The 2.4 GHz industrial, scientific, and medical (ISM) band PHY signaling techniques and interface functions that are controlled by the IEEE 802.15.1-2005 MAC. Requirements are defined for two reasons:
  - To provide compatibility between the radios used in the system.
  - To define the quality of the system.

Above the L2CAP layer may reside the Serial Cable Emulation Protocol based on ETSI TS 07.10 (RFCOMM), Service Discovery Protocol (SDP), Telephone Control Protocol specification (TCS), voice-quality channels for audio and telephony, and other network protocols. These protocols are necessary for interoperably for end-user products, but are outside the scope of this standard.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

2. Normative references

The following referenced documents are indispensable for the application of this standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

2.1 IEEE documents

IEEE Std 802®, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture.[1, 2]

IEEE Std 802.15.2™, IEEE Recommended Practice for Telecommunications and Information exchange between systems—Local and metropolitan area networks—Specific Requirements—Part 15.2: Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in Unlicensed Frequency Band.

2.2 ISO documents

ISO/IEC 3309, Information technology — Telecommunications and information exchange between systems — High-level data link control (HDLC) procedures — Frame structure.[3]

ISO/IEC 7498-1, Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model.

ISO/IEC 8802-2, Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 2: Logical link control.

ISO/IEC 10039, Information technology — Open Systems Interconnection — Local Area Networks — Medium Access Control (MAC) service definition.

ISO/IEC 15802-1, Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Common specifications — Part 1: Medium Access Control (MAC) service definition.

2.3 ITU documents

ITU-T Recommendation G.711, Pulse code modulation (PCM) of voice frequencies.[4]

ITU-T Recommendation O.150, Digital test patterns for performance measurements on digital transmission equipment.

ITU-T Recommendation O.153, Basic parameters for the measurement of error performance at bit rates below the primary rate.

---

[1]IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

[3]ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[4]ITU-T publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (http://www.itu.int/).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

ITU-T Recommendation X.200, Information technology—Open systems interconnection—Basic reference model: The basic model.

2.4 Other documents

IETF RFC 1363, A Proposed Flow Specification.[5]

IETF RFC 1661, The Point-to-Point Protocol (PPP).

IrDA Object Exchange Protocol (IrOBEX), Version 1.2.[6]

[5] IETF documents are available from Internet Engineering Task Force (http://www.ietf.org/).
[6] IrDA documents are available from the Infrared Data Association (http://www.irda.org/).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

3. Definitions

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [B7][7], should be referenced for terms not defined in this clause.

3.1 active slave broadcast (ASB): The logical transport that is used to transport Logical Link Control and Adaptation Protocol (L2CAP) user traffic to all active devices in the piconet.

3.2 ad hoc network: A network typically created in a spontaneous manner. An ad hoc network requires no formal infrastructure and is limited in temporal and spatial extent.

3.3 authenticated device: A device whose identity has been verified during the lifetime of the current link, based on the authentication procedure.

3.4 authentication: A generic procedure based on link management profile authentication that determines whether a link key exists or, on Link Manager Protocol (LMP) pairing, whether no link key exists.

3.5 authorization: A procedure where a user of a device grants a specific (remote) device access to a specific service. Authorization implies that the identity of the remote device can be verified through authentication.

3.6 authorize: The act of granting a specific device access to a specific service. It may be based upon user confirmation or given the existence of a trusted relationship.

3.7 baseband (BB): The part of the system that specifies or implements the medium access control (MAC) layer and physical layer (PHY) procedures to support the exchange of real-time voice, data information streams, and ad hoc networking between devices.

3.8 beacon train: A pattern of reserved slots within a basic or adapted piconet physical channel. Transmissions starting in these slots are used to resynchronize parked devices.

3.9 Bluetooth device address (BD_ADDR): The address used to identify a device conforming to this standard.

3.10 Bluetooth wireless technology: The general term used to describe the technolgy orginally developed by the Bluetooth Special Interest Group (SIG). It defines a wireless communication link, operating in the unlicensed industrial, scientific, and medical (ISM) band at 2.4 GHz using a frequency hopping transceiver. The link protocol is based on time slots.

3.11 bond: A relation between two devices defined by creating, exchanging, and storing a common link key. The bond is created through the bonding or Link Manager Protocol (LMP) pairing procedures.

3.12 channel: Either a physical channel or an Logical Link Control and Adaptation Protocol (L2CAP) channel, depending on the context.

3.13 connect (to service): The establishment of a connection to a service. If not already done, this also includes establishment of a physical link, logical transport, logical link, and Logical Link Control and Adaptation Protocol (L2CAP) channel.

3.14 connectable device: A device in range that periodically listens on its page scan physical channel and will respond to a page on that channel.

[7]The numbers in brackets correspond to the numbers of the bibliography in Annex A.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

3.15 connected devices: Two devices in the same piconet and with a physical link between them.

3.16 connecting: A phase in the communication between devices when a connection between them is being established. (Connecting phase follows after the link establishment phase is completed.)

3.17 connection: A connection between two peer applications or higher layer protocols mapped onto a Logical Link Control and Adaptation Protocol (L2CAP) channel.

3.18 connection establishment: A procedure for creating a connection mapped onto a channel.

3.19 controller: A subsystem containing the physical layer (PHY), baseband (BB), resource controller, link manager (LM), device manager, and a host controller interface (HCI) conforming to this standard.

3.20 coverage area: The area where two devices can exchange messages with acceptable quality and performance.

3.21 creation of a secure connection: A procedure of establishing a connection, including authentication and encryption.

3.22 creation of a trusted relationship: A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication and pairing (if the link key is not available).

3.23 device: A device that is capable of short-range wireless communications using this standard.

3.24 device address: A 48-bit address used to identify each device.

3.25 device discovery: A procedure for retrieving the device address, clock, class-of-device field, and used page scan mode from discoverable devices.

3.26 discoverable device: A device in range that periodically listens on an inquiry scan physical channel and will respond to an inquiry on that channel. Discoverable devices are normally also connectable.

3.27 estimated clock (CLKE): Estimate of another device's clock. CLKE may be a slave's estimate of a master's clock, a paging devices's estimate of the paged device's clock, or other such use.

3.28 host: A computing device, peripheral, cellular telephone, access point to public switched telephone network (PSTN) or local area network (LAN), etc. A host attached to a controller may communicate with other hosts attached to their controllers as well.

3.29 host controller interface (HCI): A command interface to the baseband (BB) controller and link manager (LM) that provides access to hardware status and control registers and provides a uniform method of accessing the BB capabilities.

3.30 idle: Description of a device, as seen from a remote device, when no link is established between the devices.

3.31 inquiring device: A device that is carrying out the inquiry procedure.

3.32 inquiry: A procedure where a device transmits inquiry messages and listens for responses in order to discover the other devices that are within the coverage area.

3.33 inquiry scan: A procedure where a device listens for inquiry messages received on its inquiry scan physical channel.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

3.34 isochronous data: Information in a stream where each information entity in the stream is bound by a time relationship to previous and successive entities.

3.35 known device: A device for which at least the Bluetooth device address (BD_ADDR) is stored.

3.36 link: Shorthand for a logical link.

3.37 link establishment: A procedure for establishing the default ACL link and hierarchy of links and channels between devices.

3.38 link key: A secret key that is known by two devices and is used in order to authenticate each device to the other.

3.39 LMP authentication: A procedure on the Link Manager Protocol (LMP) level for verifying the identity of a remote device. The procedure is based on a challenge-response mechanism using a random number, a secret key, and the Bluetooth device address (BD_ADDR) of the noninitiating device. The secret key used can be a previously exchanged link key.

3.40 LMP pairing: A procedure that authenticates two devices, based on a personal identification number (PIN), and subsequently creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection. The procedure consists of the following steps: creation of an initialization key (based on a random number and a PIN), creation and exchange of a common link key, and Link Manager Protocol (LMP) authentication based on the common link key.

3.41 logical channel: Identical to a Logical Link Control and Adaptation Protocol (L2CAP) channel, but deprecated due to inconsistent usage in IEEE Std 802.15.1-2002.

3.42 logical link: The lowest architectural level used to offer independent data transport services to clients of the system.

3.43 logical transport: Used to represent commonality between different logical links due to shared acknowledgment protocol and link identifiers.

3.44 L2CAP channel: A logical connection on the Logical Link Control and Adaptation Protocol (L2CAP) level between two devices serving a single application or higher layer protocol.

3.45 L2CAP channel establishment: A procedure for establishing a logical connection on the Logical Link Control and Adaptation Protocol (L2CAP) level.

3.46 master clock (CLK): Native clock of the piconet's master.

3.47 mode: A set of directives that defines how a device will respond to certain events.

3.48 name discovery: A procedure for retrieving the user-friendly name (the device name) of a connectable device.

3.49 native clock (CLKN): A 28-bit clock internal to a controller subsystem that ticks every 312.5 μs. The value of this clock defines the slot numbering and timing in the various physical channels.

3.50 packet: Format of aggregated bits that are transmitted on a physical channel.

3.51 page: The initial phase of the connection procedure where a device transmits a train of page messages until a response is received from the target device or a timeout occurs.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

3.52 page scan: A procedure where a device listens for page messages received on its page scan physical channel.

3.53 paging device: A device that is carrying out the page procedure.

3.54 paired device: A device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).

3.55 parked device: A device operating in a basic mode piconet that is synchronized to the master, but has given up its default ACL logical transport.

3.56 parked slave broadcast (PSB): The logical transport that is used for communications from the master to parked slave devices. These communications may also be received by active devices.

3.57 participant in multiple piconets: A device that is concurrently a member of more than one piconet. It achieves this status using time division multiplexing (TDM) to interleave its activity on each piconet physical channel.

3.58 personal identification number (PIN): A user-friendly number that can be used to authenticate connections to a device before pairing has taken place.

3.59 physical channel: A channel characterized by synchronized occupancy of a sequence of radio frequency (RF) carriers by one or more devices. A number of physical channel types exist with characteristics defined for their different purposes.

3.60 physical link: A connection on the baseband (BB) level between two devices established using paging.

3.61 piconet: A collection of devices occupying a shared physical channel where one of the devices is the piconet master and the remaining devices are connected to it.

3.62 piconet physical channel: A channel that is divided into time slots in which each slot is related to a radio frequency (RF) hop frequency. Consecutive hops normally correspond to different RF hop frequencies and occur at a standard hop rate of 1600 hop/s. These consecutive hops follow a pseudo-random hopping sequence, hopping through a 79-RF channel set, or optionally fewer channels when adaptive frequency hopping (AFH) is in used.

3.63 piconet master: The device in a piconet whose clock and device address are used to define the piconet physical channel characteristics.

3.64 piconet slave: Any device in a piconet that is not the piconet master, but is connected to the piconet master, and that controls piconet timing and access by its transmissions to slaves.

3.65 prepaired device: A device with which a link key was exchanged and stored before link establishment.

3.66 scatternet: Two or more piconets that include one or more devices participating in more than one piconet.

3.67 service discovery (SD): Procedures for querying and browsing for services offered by or through another device.

3.68 service layer protocol: A protocol that uses a Logical Link Control and Adaptation Protocol (L2CAP) channel for transporting protocol data units (PDUs).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

3.69 silent device: A device appears as silent to a remote device if it does not respond to inquiries made by the remote device.

3.70 trusted device: A paired device that is explicitly marked as trusted.

3.71 unknown device: A device for which no information (e.g., device address, link key) is stored.

3.72 unpaired device: A device for which there was no exchanged link key available before connection establishment was requested.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

4. Acronyms and abbreviations

This clause contains two classes of acronyms and abbreviations. The first class is based on this and other standards and is the type usually found in IEEE standards. The second class refers to acronyms and abbreviations from the Bluetooth specification that are used by this standard. This second class is included in this standard as an aid to the reader.

4.1 Standard-based acronyms and abbreviations

ACK	acknowledge
ACL	asynchronous connection-oriented [logical transport]
ACL-C	ACL control [logical link] (LMP)
ACL-U	ACL user [logical link] (L2CAP)
ACO	authenticated ciphering offset
AFH	adaptive frequency hopping
AHS	adapted hop sequence
AR_ADDR	access request address
ARQ	automatic repeat request
ARQN	acknowledgment indication
ASB	active slave broadcast [logical transport]
ASB-U	active slave broadcast user [logical link] (L2CAP)
BB	baseband
BCH	Bose, Chaudhuri, and Hocquenghem
BD_ADDR	device address
BER	bit error rate
BT	bandwidth time
CAC	channel access code
CID	channel identifier
CL	connectionless
CLK	master clock
CLKE	estimated clock
CLKN	native clock
CODEC	coder decoder
COF	ciphering offset
CQDDR	channel quality-driven data rate
CRC	cyclic redundancy check
CVSD	continuous variable slope delta [modulation]
DA	destination address [field]

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

DAC	device access code
DCI	default check initialization
DH	data–high rate [packet]
DIAC	dedicated inquiry access code
DLL	data link layer
DM	data–medium rate [packet]
DUT	device under test
DV	data-voice [packet]
EN_RAND	encryption random number
ERP	ear reference point
eSCO	extended synchronous connection-oriented [logical transport]
eSCO-S	stream extended synchronous connection-oriented
EV	extended voice [packet]
ERTX	extended response timeout expired [timer]
FCS	frame check sequence
FEC	forward error correction
FHS	frequency hop synchronization
FHSS	frequency hopping spread spectrum
FIFO	first in first out
GAP	generic access profile
GFSK	Gaussian frequency shift keying
GIAC	general inquiry access code
HCI	host controller interface
HEC	header error check
HID	human interface device
HV	high-quality voice [packet]
IAC	inquiry access code
IN_RAND	initialization random number
IP	Internet Protocol
ISDN	integrated services digital network
ISM	industrial, scientific, and medical
L2CAP	Logical Link Control and Adaptation Protocol
LAP	lower address part
LC	link control [logical link]
LCID	local channel identifier
LCP	Link Control Protocol

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

LFSR	linear feedback shift register
LIAC	limited inquiry access code
LLC	logical link control
LLID	logical link identifier
LM	link manager
LMP	Link Manager Protocol
LPO	low-power oscillator
LR	loudness rating
LSB	least significant bit
LT_ADDR	logical transport address
M	master or mandatory
MAC	medium access control
MPS	maximum PDU payload size
MSB	most significant bit
MSC	message sequence chart
MSDU	MAC service data unit
MTU	maximum transmission unit
NAK	negative acknowledge
NAP	nonsignificant address part
NOP	no operation
O	optional
OBEX	Object Exchange Protocol
OCF	opcode command field
OGF	opcode group field
PCM	pulse code modulation
PDU	protocol data unit
PGA	programmable gain amplifier
PHT	pseudo-Hadamard transform
PIN	personal identification number
PM_ADDR	parked member address
PN	pseudo-random noise
POS	personal operating space
ppm	parts per million
PRBS	pseudo-random bit sequence
PSB	parked slave broadcast [logical transport]
PSB-C	parked slave broadcast control [logical link] (LMP)

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

PSB-U	parked slave broadcast user [logical link] (L2CAP)
PSM	protocol/service multiplexer
PSTN	public switched telephone network
QoS	quality of service
RAND	random number
RF	radio frequency
RFCMode	retransmission and flow control mode
RFCOMM	Serial Cable Emulation Protocol based on ETSI TS 07.10
RLR	receive loudness rating
RSSI	received signal strength indication
RX	receive
RTX	response timeout expired [timer]
S	slave
SA	source address [field]
SAP	service access point
SAR	segmentation and reassembly
SCO	synchronous connection-oriented [logical transport]
SCO-S	stream synchronous connection-oriented (unframed)
SD	service discovery
SDAP	service discovery applicaiton profile
SDP	Service Discovery Protocol
SDU	service data unit
SEQN	sequence number
SLR	send loudness rating
SRES	signed response
TCS	Telephony Control Protocol specification
TDD	time-division duplex
TDM	time-division multiplexing
TX	transmit
UAP	upper address part
u_int	unsigned integer
USB	universal serial bus

4.2 Bluetooth specification names

References to upper layer Bluetooth protocols use the abbreviations in Table 1.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 1—Abbreviations of the Bluetooth specification names

Name	Reference	Placement in Bluetooth specification
A2DP	Advanced Audio Distribution Profile Specification	vol 10 part C
AVCTP	A/V Control Transport Protocol Specification	vol 10 part F
AVDTP	A/V Distribution Transport Profile Specification	vol 10 part A
AVRCP	A/V Remote Control Profile Specification	vol 10 part G
BB	Baseband Specification	vol 2 part B
BIP	Basic Imaging Profile	vol 8 part E
BNEP	Bluetooth Network Encapsulation Protocol Specification	vol 6 part A
BPP	Basic Printing Profile Specification	vol 8 part F
CIP	Common Integrated Services Digital Network (ISDN) Access Profile Specification	vol 12 part A
CTP	Cordless Telephony Profile Specification	vol 9 part B
DUN	Dial-Up Networking Profile Specification	vol 7 part C
ESDP / UPNP	Extended Service Discovery Profile	vol 6 part D
FAX	Fax Profile Specification	vol 7 part D
FTP	File Transfer Profile Specification	vol 8 part C
GAP	Generic Access Profile Specification	vol 3 part C
GAVDP	Generic A/V Distribution Profile Specification	vol 10 part B
GOEP	Generic Object Exchange Profile Specification	vol 8 part A
HCI (1)	Host Controller Interface Functional Specification	vol 2 part E
HCI (2)	Host Controller Interface Transport Layers Specification	vol 4 part A-C
HCRP	Hardcopy Cable Replacement Profile Specification	vol 11 part B
HFP	Hands-Free Profile Specification	vol 7 part E
HID	Human Interface Device Profile Specification	vol 11 part A
HSP	Headset Profile Specification	vol 7 part F
ICP	Intercom Profile Specification	vol 9 part C
L2CAP	Logical Link Control and Adaptation Protocol Specification	vol 3 part A
LAP	LAN Access Profile Specification	deprecated
LMP	Link Manager Protocol Specification	vol 2 part C
MSC	Message Sequence Charts	vol 2 part F
OPP	Object Push Profile Specification	vol 8 part B
PAN	Personal Area Networking Profile Specification	vol 6 part B
RF	Radio Specification	vol 2 part A

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 1—Abbreviations of the Bluetooth specification names *(continued)*

Name	Reference	Placement in Bluetooth specification
RFCOMM	Serial Cable Emulation Protocol based on ETSI TS 07.10	vol 7 part A
SAP	SIM Access Profile Specification	vol 12 part C
SDAP	Service Discovery Application Profile Specification	vol 5 part B
SDP (1)	Service Discovery Protocol Specification (server)	vol 3 part B
SDP (2)	Service Discovery Protocol Specification (client)	vol 5 part A
SPP	Serial Port Profile Specification	vol 7 part B
Synch	Synchronization Profile Specification	vol 8 part D
TCI	Test Control Interface	vol 3 part D, section 2
TCP	Telephony Control Protocol Specification	vol 9 part A
UDI	Unrestricted Digital Information Profile Specification	vol 12 part B

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

5. General description

5.1 New features

Several new features are introduced in IEEE Std 802.15.1-2005. The major areas of improvement are as follows:

— Architectual overview
— Faster connection
— Adaptive frequency hopping (AFH)
— Extended SCO links
— Enhanced error detection and flow control
— Enhanced synchronization capability
— Enhanced flow specification

These feature descriptions are incorporated into the text within this standard.

5.2 Changes in wording

Two general classes of changes to the wording of IEEE Std 802.15.1-2002 have been done in IEEE Std 802.15.1-2005. They are a conformance to the formalization of the language by using conventions established by the IEEE and a regularization of Bluetooth wireless technology-specific terms.

5.2.1 IEEE language update

Many portions of IEEE Std 802.15.1-2002 used imprecise or inaccurate terms to describe attributes of the protocol. This standard now conforms to the correct usage of the key verbs that describe requirements. Table 2 is a summary of the verbs whose usage was regularized based on the “IEEE Style Guide” [B8].

Table 2—IEEE nomenclature

shall	is required to – used to define requirements
must	is a natural consequence of – used only to describe unavoidable situations
will	it is true that – used only in statements of fact
should	is recommended that – used to indicate that among several possibilities one is recommended as particularly suitable, but not required
may	is permitted to – used to allow options
can	is able to – used to relate statements in a causal fashion
is	is defined as – used to further explain elements that are previously required or allowed
note	<informational text only>

5.2.2 Nomenclature changes

The nomenclature used to describe the protocol has also been changed in IEEE Std 802.15.1-2005. Several terms were used more than once, for different concepts in IEEE Std 802.15.1-2002. The text has been updated to regularize this standard-specific usage. The nomenclature is introduced together with the new features in the new architecture subclause (see 6.2).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

5.3 Structure changes

This standard has been significantly restructured for better consistency and readability. The most important structure changes have been performed in BB, Link Manager Protocol (LMP), host controller interface (HCI), and L2CAP. The text in these clauses have been rearranged to provide the following:

— Presentation of the information in a more logical progression
— Removal of redundant text and requirements
— Consolidation of BB-related requirements (e.g., moving the BB timers and audio subclauses into Clause 8 about the BB)

5.4 Deprecated features

As this standard and the Bluetooth specification continue to evolve, some features, protocols, and profiles are replaced with new ways of performing the same function. Often these changes reflect the evolution of the communications industry. Some of the changes merely reflect an evolved understanding of the WPAN environment itself.

The functions no longer recommended are being deprecated. The term *deprecation* does not mean that these functions are no longer allowed, but that they are no longer recommended as the best way of performing a given function.

Features deprecated in IEEE Std 802.15.1-2005 are as follows:

— The use of unit keys for security
— Optional paging schemes
— The 23-channel hopping sequence

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

6. Architecture

This standard is a formalization of Bluetooth wireless technology, a short-range communications system intended to replace the cable(s) connecting portable and/or fixed electronic devices. Key features are robustness, low power, and low cost. Many features of the core specification are optional, allowing product differentiation.

The term *core system* is used in this clause to denote the combination of a radio frequency (RF) transceiver, BB, and protocol stack. The system offers services that enable the connection of devices and the exchange of a variety of classes of data between these devices.

This clause of this standard provides an overview of the system architecture, communication topologies, and data transport features. This clause is informative.

6.1 General description

The RF (PHY) operates in the unlicensed ISM band at 2.4 GHz. The system employs a frequency hop transceiver to combat interference and fading and provides many frequency hopping spread spectrum (FHSS) carriers. RF operation uses a shaped, binary frequency modulation to minimize transceiver complexity. The symbol rate is 1 Msymbol/s supporting the bit rate of 1 Mb/s.

During typical operation, a physical radio channel is shared by a group of devices that are synchronized to a common clock and frequency hopping pattern. One device provides the synchronization reference and is known as the *master*. All other devices are known as *slaves*. A group of devices synchronized in this fashion form a *piconet*. This is the fundamental form of communication in the technology.

Devices in a piconet use a specific frequency hopping pattern, which is algorithmically determined by fields in the device address and the clock of the master. The basic hopping pattern is a pseudo-random ordering of the 79 frequencies in the ISM band. The hopping pattern may be adapted to exclude a portion of the frequencies that are used by interfering devices. The adaptive hopping technique improves coexistence with static (nonhopping) ISM systems when these are collocated and implements some of the recommendations of IEEE Std 802.15.2-2003.

The physical channel is subdivided into time units known as *slots*. Data are transmitted between devices in packets, which are positioned in these slots. When circumstances permit, a number of consecutive slots may be allocated to a single packet. Frequency hopping takes place between the transmission or the reception of packets. This standard provides the effect of full duplex transmission through the use of a time-division duplex (TDD) scheme.

Above the physical channel, there is a layering of links and channels and associated control protocols. The hierarchy of channels and links from the physical channel upwards is physical channel, physical link, logical transport, logical link, and L2CAP channel. These are discussed in more detail in 6.4.4 through 6.5, but are introduced here to aid the understanding of the remainder of this clause.

Within a physical channel, a physical link is formed between any two devices that transmit packets in either direction between them. In a piconet physical channel, there are restrictions on which devices may form a physical link. There is a physical link between each slave and the master. Physical links are not formed directly between the slaves in a piconet.

The physical link is used as a transport for one or more logical links that support unicast synchronous, asynchronous and isochronous traffic, and broadcast traffic. Traffic on logical links is multiplexed onto the physical link by occupying slots assigned by a scheduling function in the resource manager.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A control protocol for the BB layer and PHY is carried over logical links in addition to user data. This is the LMP. Devices that are active in a piconet have a default asynchronous connection-oriented (ACL) logical transport that is used to transport the LMP signalling. For historical reasons, this is referred to as the ACL logical transport. The default ACL logical transport is the one that is created whenever a device joins a piconet. Additional logical transports may be created to transport synchronous data streams when this is required.

The LM function uses LMP to control the operation of devices in the piconet and provide services to manage the lower architectural levels (i.e., PHY and BB). The LMP is carried only on the default ACL logical transport and the default broadcast logical transport.

Above the BB, L2CAP provides a channel-based abstraction to applications and services. It carries out segmentation and reassembly (SAR) of application data and multiplexing and demultiplexing of multiple channels over a shared logical link. L2CAP has a protocol control channel that is carried over the default ACL logical transport. Application data submitted to the L2CAP may be carried on any logical link that supports the L2CAP.

6.2 Core system architecture

The core system covers the four lowest segments and associated protocols defined by this standard, and the overall profile requirements are specified in the generic access profile (GAP) (see Annex B). A complete application generally requires a number of additional service and higher layer protocols that are defined in the Bluetooth specification and are not described in this standard. The core system architecture is shown in Figure 1.

Core system architecture shows the four lowest layers, each with its associated communication protocol. The lowest three layers are sometimes grouped into a subsystem (known as the *controller*). This is a common implementation involving a standard physical communications interface (i.e., the host controller interface or HCI) and remainder of the system. This includes the L2CAP, service, and higher layers (known as the *host*). Although this interface is optional, the architecture is designed to allow for its existence and characteristics. This standard enables interoperability between independent systems by defining the protocol messages exchanged between equivalent layers and also interoperability between independent subsystems by defining a common interface between controllers and hosts.

A number of functional blocks are shown in Figure 1 and the path of services and data between these. The functional blocks shown in the diagram are informative; in general, this standard does not define the details of implementations except where this is required for interoperability. Thus the functional blocks in Figure 1 are shown in order to aid description of the system behavior. An implementation may be different from the system shown in Figure 1.

Standard interactions are defined for all interdevice operation, where devices exchange protocol signalling according to this standard. The core system protocols are the Radio Frequency (RF) Protocol, Link Control Protocol (LCP), LMP, and L2CAP, all of which are fully defined in subsequent parts of this standard.

The core system offers services through a number of service access points (SAPs) that are shown in Figure 1 as ellipses. These services consist of the basic primitives that control the core system. The services can be split into three types:

— Device control services that modify the behavior and modes of a device
— Transport control services that create, modify, and release traffic bearers (channels and links)
— Data services that are used to submit data for transmission over traffic bearers

It is common to consider the first two as belonging to the C-plane and the last as belonging to the U-plane.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

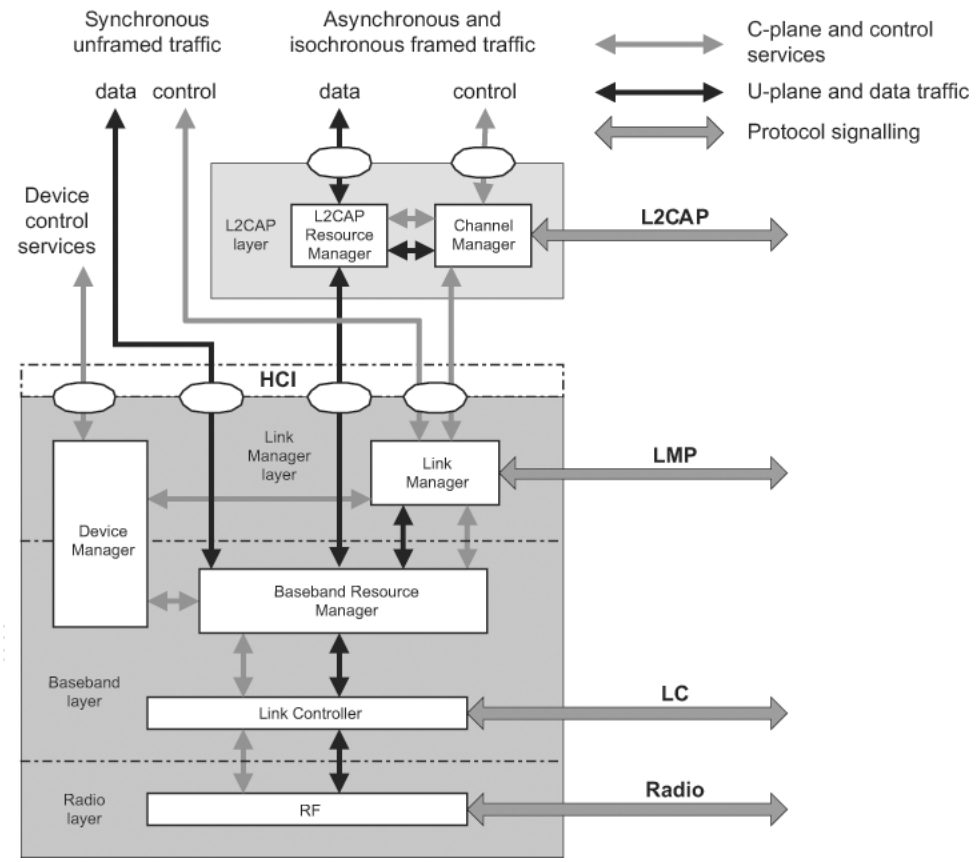


Figure 1—Core system architecture

A service interface to the controller subsystem is defined so that the controller may be considered a standard part. In this configuration, the controller operates the lowest three layers, and L2CAP is contained with the rest of the application in a host system. This standard interface is called the host controller interface (HCI), and its SAPs are represented by the ellipses on the upper edge of the controller subsystem in Figure 1. Implementation of this standard service interface is optional.

As the architecture is defined with the possibility of separate host and controller communicating through an HCI, a number of general assumptions are made. The controller is assumed to have limited data buffering capabilities in comparison with the host. Therefore, L2CAP is expected to carry out some simple resource management when submitting L2CAP protocol data units (PDUs) to the controller for transport to a peer device. This includes segmentation of L2CAP service data units (SDUs) into more manageable PDUs and then the fragmentation of PDUs into start and continuation packets of a size suitable for the controller buffers, and management of the use of controller buffers to ensure availability for channels with quality of service (QoS) commitments.

The BB protocol provides the basic ARQ Protocol. The L2CAP can optionally provide a further error detection and retransmission to the L2CAP PDUs. This feature is recommended for applications with requirements for a low probability of undetected errors in the user data. A further optional feature of L2CAP is a window-based flow control that can be used to manage buffer allocation in the receiving device. Both of these optional features augment the QoS performance in certain scenarios.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

6.3 Core architectural blocks

This subclause describes the function and responsibility of each of the blocks shown in Figure 1, which describes a possible implementation architecture. An implementation is not required to follow the architecture described in this clause.

6.3.1 Channel manager

The channel manager is responsible for creating, managing, and destroying L2CAP channels for the transport of service protocols and application data streams. The channel manager uses the L2CAP to interact with a channel manager on a remote (peer) device to create these L2CAP channels and connect their endpoints to the appropriate entities. The channel manager interacts with its local LM to create new logical links (if necessary) and to configure these links to provide the required QoS for the type of data being transported.

6.3.2 L2CAP resource manager

The L2CAP resource manager block is responsible for managing the ordering of submission of PDU fragments to the BB and some relative scheduling between channels to ensure that L2CAP channels with QoS commitments are not denied access to the physical channel due to controller resource exhaustion. This is required because the architectural model does not assume that the controller has limitless buffering or that the HCI is a pipe of infinite bandwidth.

L2CAP resource managers may also carry out traffic conformance policing to ensure that applications are submitting L2CAP SDUs within the bounds of their negotiated QoS settings. The general data transport model assumes well-behaved applications and does not define how an implementation is expected to deal with this problem.

6.3.3 Device manager

The device manager is the functional block in the BB that controls the general behavior of the device. It is responsible for all operation of the system that is not directly related to data transport. This includes functions such as inquiring for the presence of other nearby devices, connecting to other devices, or making the device discoverable or connectable by other devices.

The device manager requests access to the transport medium from the BB resource controller in order to carry out its functions.

The device manager also controls local device behavior implied by a number of the HCI commands, such as managing the device’s local name, any stored link keys, and other functionality.

6.3.4 Link manager (LM)

The LM is responsible for the creation, modification, and release of logical links (and, if required, their associated logical transports) as well as the update of parameters related to physical links between devices. The LM achieves this by communicating with the LM in remote devices using the LMP.

LMP allows the creation of new logical links and logical transports between devices when required as well as the general control of link and transport attributes such as the enabling of encryption on the logical transport, the adapting of transmit power on the physical link, or the adjustment of QoS settings for a logical link.

6.3.5 BB resource manager

The BB resource manager is responsible for all access to the PHY. It has two main functions. At its heart is a scheduler that grants time on the physical channels to all of the entities that have negotiated an access

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

contract. The other main function is to negotiate access contracts with these entities. An access contract is effectively a commitment to deliver a certain QoS that is required in order to provide a user application with an expected performance.

The access contract and scheduling function must take account of any behavior that requires use of the radio, e.g., the normal exchange of data between connected devices over logical links, the logical transports, the carrying out of inquiries, the making of connections, the state of being discoverable or connectable, or the taking of readings from unused carriers during the use of AFH mode.

In some cases, the scheduling of a logical link results in changing to a different physical channel from the one that was previously used. This may be, for example, due to involvement in scatternet, a periodic inquiry function, or page scanning. When the physical channels are not time-slot-aligned, then the resource manager also accounts for the realignment time between slots on the original physical channel and slots on the new physical channel. In some cases, the slots will be naturally aligned due to the same device clock being used as a reference for both physical channels.

6.3.6 Link controller

The link controller is responsible for the encoding and decoding of packets from the data payload and parameters related to the physical channel, logical transport, and logical link.

The link controller carries out the LCP signalling (in close conjunction with the scheduling function of the resource manager), which is used to communicate flow control and acknowledgment and retransmission request signals. The interpretation of these signals is a characteristic of the logical transport associated with the BB packet. Interpretation and control of the link control signalling is normally associated with the resource manager's scheduler.

6.3.7 Radio frequency (RF)

The RF block is responsible for transmitting and receiving packets of information on the physical channel. A control path between the BB and the RF block allows the BB block to control the timing and frequency carrier of the RF block. The RF block transforms a stream of data to and from the physical channel and the BB into required formats.

6.4 Data transport architecture

The data transport system follows a layered architecture. This explanation of the system describes the core transport layers up to and including L2CAP channels. All operational modes follow the same generic transport architecture, which is shown in Figure 2.

For efficiency and legacy reasons, the transport architecture includes a subdivision of the logical layer, distinguishing between logical links and logical transports. This subdivision provides a general (and commonly understood) concept of a logical link that provides an independent transport between two or more devices. The logical transport sublayer is required to describe the interdependence between some of the logical link types, mainly for reasons of legacy behavior.

IEEE Std 802.15.1-2002 described the ACL and SCO links as physical links. With the addition of extended SCO (eSCO) and for future expansion, it is better to consider these as logical transport types, which more accurately encapsulates their purpose. However, they are not as independent as might be desired, due to the shared use of the logical transport address (LT_ADDR) between SCO and ACL. Hence, the architecture is incapable of representing these logical transports with a single transport layer. The additional logical transport layer goes some way toward describing this behavior.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

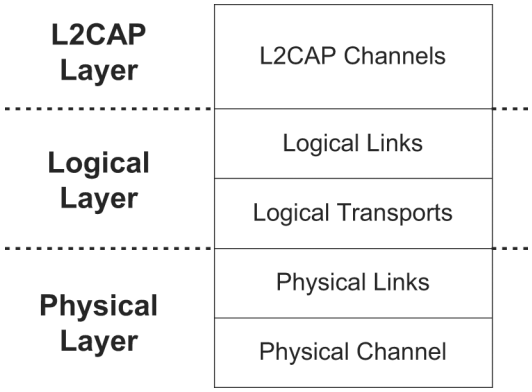


Figure 2—Generic data transport architecture

6.4.1 Core traffic bearers

The core system provides a number of standard traffic bearers for the transport of service protocol and application data. These are shown in Figure 3. For ease of representation, this is shown with higher layers to the left and lower layers to the right.

The core traffic bearers that are available to applications are shown in Figure 3 as the shaded rounded rectangles. The architectural layers that are defined to provide these services are described in 6.2. A number of data traffic types are shown on the left of the figure as linked to the traffic bearers that are typically suitable for transporting that type of data traffic.

The logical links are identified using the names of the associated logical transport and a suffix that indicates the type of data that is transported. The letter “C” indicates control links carrying LMP messages. The letter U indicates L2CAP links carrying user data (L2CAP PDUs). The letter S indicates stream links carrying unformatted synchronous or isochronous data. It is common for the suffix to be removed from the logical link without introducing ambiguity; thus, a reference to the default ACL logical transport can be resolved to mean the ACL-C logical link in cases where the LMP is being discussed or the ACL-U logical link when the L2CAP is being discussed.

The mapping of application traffic types to core traffic bearers in Figure 3 is based on matching the traffic characteristics with the bearer characteristics. It is recommended to use these mappings as they provide the most natural and efficient method of transporting the data with its given characteristics.

However, an application—or an implementation of the core system—may choose to use a different traffic bearer or a different mapping to achieve a similar result. For example, in a piconet with only one slave, the master may choose to transport L2CAP broadcasts over the ACL-U logical link rather than over the active slave broadcast user (ASB-U) or parked slave broadcast user (PSB-U) logical links. This will probably be more efficient in terms of bandwidth if the physical channel quality is not degraded. Use of alternative transport paths to those in Figure 3 is acceptable only if the characteristics of the application traffic type are preserved.

Figure 3 shows a number of application traffic types. These are used to classify the types of data that may be submitted to the core system. The original data traffic type may not be the same as the type that is submitted to the core system if an intervening process modifies it. For example, video data are generated at a constant rate, but an intermediate coding process may alter this to a variable rate, e.g., by MPEG4 encoding. For the purposes of the core system, only the characteristic of the submitted data is of interest.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

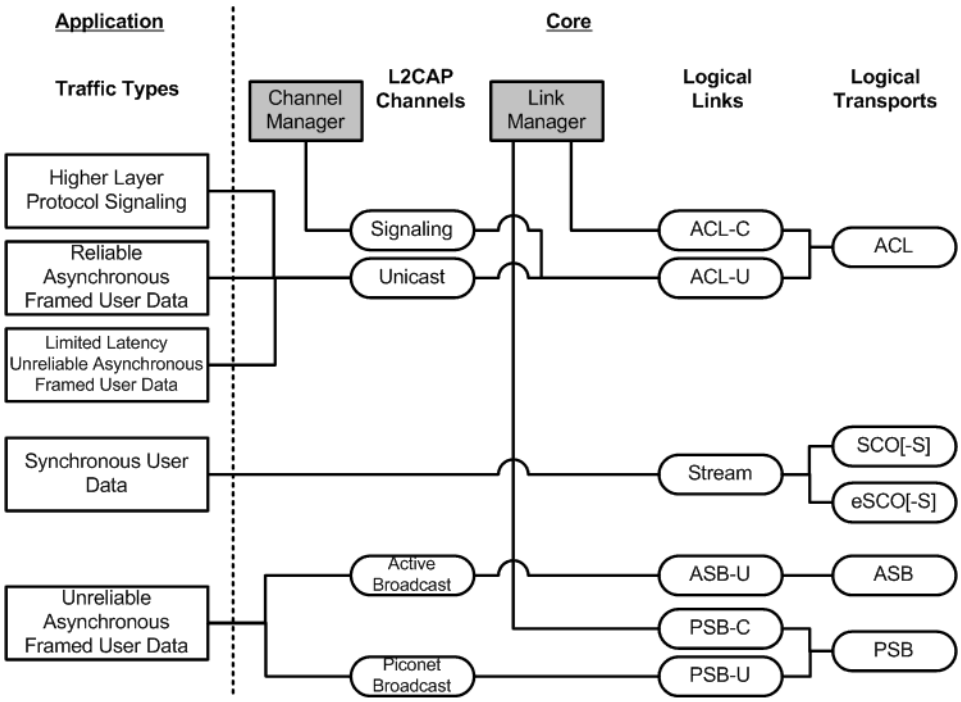


Figure 3—Traffic bearers

6.4.1.1 Framed data traffic

The L2CAP services provide a frame-oriented transport for asynchronous and isochronous user data. The application submits data to this service in variable-sized frames—up to a negotiated maximum for the channel—and these frames are delivered in the same form to the corresponding application on the remote device. There is no requirement for the application to insert additional framing information into the data, although it may do so if this is required. Such framing is invisible to the core system.

Connection-oriented L2CAP channels may be created for transport of unicast (point-to-point) data between two devices. A connectionless L2CAP channel exists for broadcasting data. In the case of piconet topologies, the master device is always the source of broadcast data, and the slave device(s) are the recipients. Traffic on the broadcast L2CAP channel is unidirectional. Unicast L2CAP channels may be unidirectional or bidirectional.

L2CAP channels have an associated QoS setting that defines constraints on the delivery of the frames of data. These QoS settings may be used to indicate, for example, that the data are isochronous and, therefore, have a limited lifetime after which they become invalid, or that the data should be delivered within a given time period, or that the data are reliable and should be delivered without error, however long this takes.

The L2CAP channel manager is responsible for arranging to transport the L2CAP channel data frames on an appropriate BB logical link, possibly multiplexing this onto the BB logical link with other L2CAP channels with similar characteristics.

6.4.1.2 Unframed data traffic

If the application does not require delivery of data in frames, possibly because it includes in-stream framing or because the data are a pure stream, then it may avoid the use of L2CAP channels and make direct use of a BB logical link. Unframed streams use SCO logical transports.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The core system supports the direct transport of application data that are isochronous and of a constant bit rate, using a stream SCO (SCO-S) or stream extended SCO (eSCO-S) logical link. These logical links reserve physical channel bandwidth and provide a constant rate transport locked to the piconet clock. Data are transported in fixed size packets at fixed intervals with both of these parameters negotiated during channel establishment. eSCO links provide a greater choice of bit rates and also provide greater reliability by using limited retransmission in case of error. SCO and eSCO logical transports do not support multiplexed logical links or any further layering within the core. An application may choose to layer a number of streams within the submitted SCO/eSCO stream, provided that the submitted stream is, or has the appearance of being, a constant rate stream.

The application chooses the most appropriate type of logical link from those available at the BB, creates and configures it to transport the data stream, and releases it when completed. The application will normally also use a framed L2CAP unicast channel to transport its C-plane information to the peer application on the remote device.

If the application data are of a variable rate (asynchronous), then they may be carried only by an L2CAP channel and hence will be treated as framed data.

6.4.1.3 Reliability of traffic bearers

This standard defines a wireless communications system. In high RF-noise environments, RF systems are inherently unreliable. To counteract this, the system provides levels of protection at each layer. The BB packet header uses forward error correction (FEC) coding to allow error correction by the receiver and a header error check (HEC) to detect errors remaining after correction. Certain BB packet types include FEC for the payload. Furthermore, some BB packet types include a cyclic redundancy check (CRC).

On ACL logical transports, the results of the error detection algorithm are used to drive a simple ARQ Protocol. This provides an enhanced reliability by retransmitting packets that do not pass the receiver's error checking algorithm. It is possible to modify this scheme to support latency-sensitive packets by discarding an unsuccessfully transmitted packet at the transmitter if the packet's useful life has expired. eSCO links use a modified version of this scheme to improve reliability by allowing a limited number of retransmissions.

The resulting reliability gained by this ARQ scheme is only as dependable as the ability of the HEC and CRC codes to detect errors. In most cases, this is sufficient; however, it has been shown that, for the longer packet types, the probability of an undetected error is too high to support typical applications, especially those with a large amount of data being transferred.

The L2CAP provides an additional level of error control that is designed to detect the occasional undetected errors in the BB protocol and request retransmission of the affected data. This provides the level of reliability required by typical IEEE 802.15.1-2005 applications.

Broadcast links have no feedback route and are unable to use the ARQ scheme, although the receiver is still able to detect errors in received packets. Instead, each packet is transmitted several times in the hope that the receiver is able to receive at least one of the copies successfully. Despite this approach there are still no guarantees of successful receipt; therefore, these links are considered unreliable.

In summary, if a link or channel is characterized as reliable, this means that the receiver is capable of detecting errors in received packets and requesting retransmission until the errors are removed. Due to the error detection system used, some residual (undetected) errors may still remain in the received data. For L2CAP channels, the level of these is comparable to other communication systems, although for logical links the residual error level is somewhat higher.

The transmitter may remove packets from the transmit queue so that the receiver does not receive all the packets in the sequence. If this happens, detection of the missing packets is delegated to the L2CAP.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

On an unreliable link, the receiver is capable of detecting errors in received packets, but cannot request retransmission. The packets passed on by the receiver may be without error, but there is no guarantee that all packets in the sequence are received. Hence, the link is considered fundamentally unreliable. There are limited uses for such links, and these uses are normally dependent on the continuous repetition of data from the higher layers while it is valid.

Stream links have a reliability characteristic somewhere between a reliable and an unreliable link, depending on the current operating conditions.

6.4.2 Transport architecture entities

The transport architecture entities are shown in Figure 4 and are described from the lowest layer upward in the subsequent subclauses.

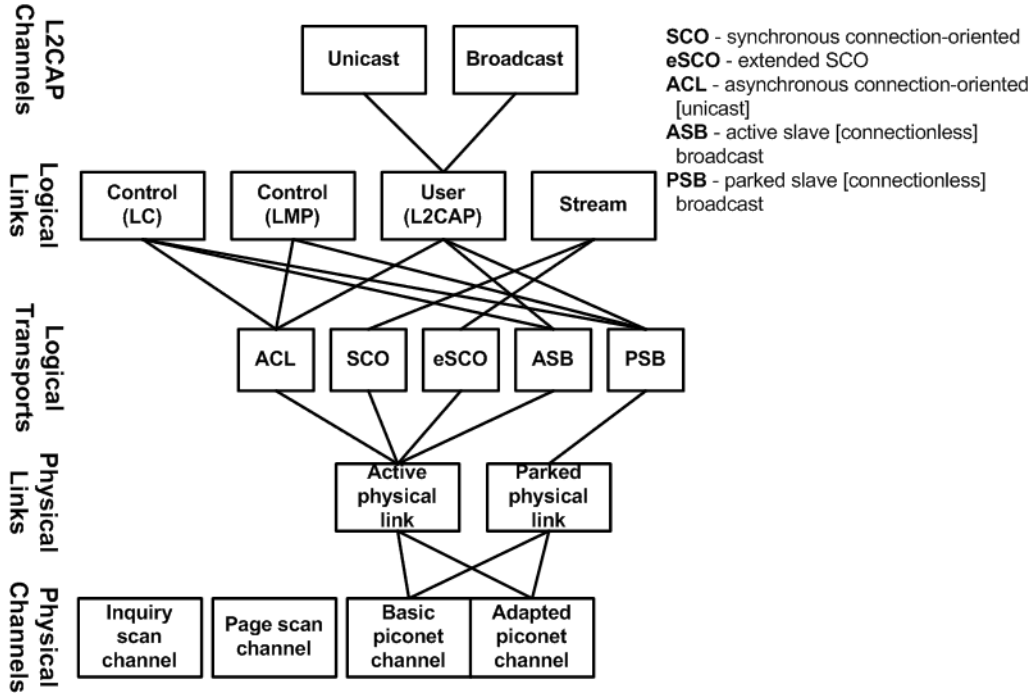


Figure 4—Overview of transport architecture entities and hierarchy

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

6.4.3 Generic packet structure

The general packet structure nearly reflects the architectural layers found in this standard. The packet structure is designed for optimal use in normal operation. It is shown in Figure 5.

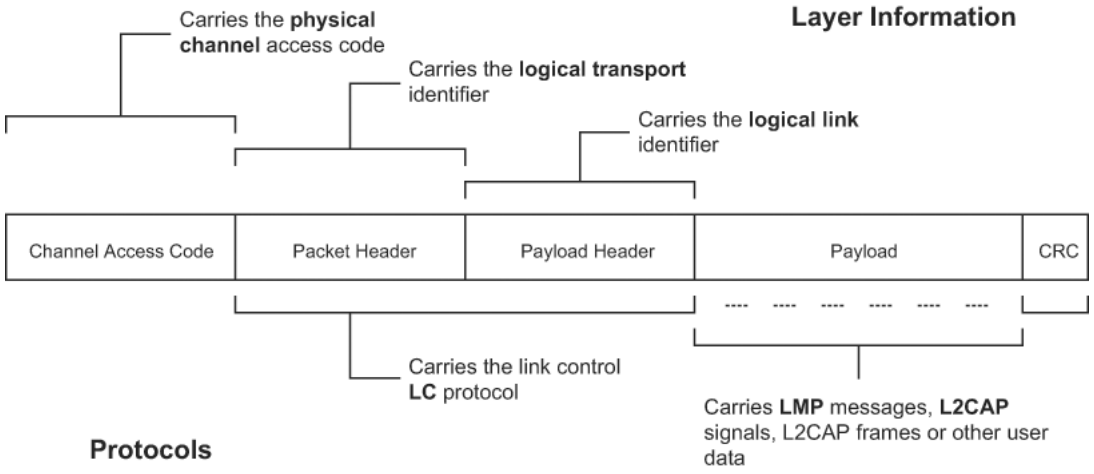


Figure 5—Packet structure

Packets normally include only the fields that are necessary to represent the layers required by the transaction. Thus a simple inquiry request over an inquiry scan physical channel does not create or require a logical link or higher layer and, therefore, consists only of the channel access code (CAC) associated with the physical channel. General communication within a piconet uses packets that include all of the fields, as all of the architectural layers are used.

All packets include the CAC. This is used to identify communications on a particular physical channel and to exclude or ignore packets on a different physical channel that happens to be using the same RF carrier in physical proximity.

There is no direct field within the packet structure that represents or contains information relating to physical links. This information is implied in the LT_ADDR carried in the packet header.

Most packets include a packet header. The packet header is always present in packets transmitted on physical channels that support physical links, logical transports, and logical links. The packet header carries the LT_ADDR, which is used by each receiving device to determine whether the packet is addressed to the device and is used to route the packet internally.

The packet header also carries part of the LCP that is operated per logical transport (except for SCO logical transports, which are not affected by the LCP bits in the packet header).

The payload header is present in all packets on logical transports that support multiple logical links. The payload header includes a logical link identifier (LLID) field used for routing the payload and a field indicating the length of the payload. Some packet types also include a CRC after the packet payload that is used to detect most errors in received packets.

The packet payload is used to transport the user data. The interpretation of these data is dependent on the logical transport and LLIDs. For ACL logical transports, LMP messages and L2CAP signals are transported in the packet payload, along with general user data from applications. For SCO and eSCO logical transports, the payload contains the user data for the logical link.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

6.4.4 Physical channels

The lowest architectural layer in the system is the physical channel. A number of types of physical channel are defined. All physical channels are characterized by an RF combined with temporal parameters and restricted by spatial considerations. All physical channel frequency hopping changes frequency periodically (frequency hop) to reduce the effects of interference and for regulatory reasons.

Two devices use a shared physical channel for communication. To achieve this, their transceivers need to be tuned to the same RF at the same time, and they need to be within a nominal range of each other.

Given that the number of RF carriers is limited and that many devices may be operating independently within the same spatial and temporal area, there is a strong likelihood of two independent devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision. To mitigate the unwanted effects of this collision, each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This CAC is a property of the physical channel. The access code is always present at the start of every transmitted packet.

Four physical channels are defined. Each is optimized and used for a different purpose. Two of these physical channels (i.e., the basic and adapted piconet physical channels) are used for communication between connected devices and are associated with a specific piconet. The remaining physical channels are used for discovering devices (i.e., the inquiry scan physical channel) and for connecting devices (i.e., the page scan physical channel).

A device can use only one of these physical channels at any given time. In order to support multiple concurrent operations, the device uses time-division multiplexing (TDM) between the channels. In this way, a device can appear to operate simultaneously in several piconets as well as being discoverable and connectable.

Whenever a device is synchronized to the timing, frequency, and access code of a physical channel, it is said to be *connected* to this channel (whether or not it is actively involved in communications over the channel). This standard assumes that a device is capable of connecting to only one physical channel at any time. Advanced devices may be capable of connecting simultaneously to more than one physical channel, but this standard does not assume that this is possible.

6.4.4.1 Basic piconet physical channel

The basic piconet physical channel is used for communication between connected devices.

The basic piconet physical channel is characterized by a pseudo-random sequence hopping through the RF channels. The hopping sequence is unique for the piconet and is determined by the device address of the master. The phase in the hopping sequence is determined by the native clock (CLKN) of the master. All devices participating in the piconet are time- and hop-synchronized to the channel.

The channel is divided into time slots where each slot corresponds to an RF hop frequency. Consecutive hops correspond to different RF hop frequencies. The time slots are numbered according to the CLKN of the piconet master. Packets are transmitted by devices participating in the piconet aligned to start at a slot boundary. Each packet starts with the channel's access code, which is derived from the device address of the piconet.

On the basic piconet physical channel, the master controls access to the channel. The master starts its transmission in even-numbered time slots only. Packets transmitted by the master are aligned with the slot start and define the piconet timing. Packets transmitted by the master may occupy up to five time slots depending on the packet type.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Each master transmission is a packet carrying information on one of the logical transports. Slave devices may transmit on the physical channel in response. The characteristics of the response are defined by the logical transport that is addressed.

For example, on the ACL logical transport, the addressed slave device responds by transmitting a packet containing information for the same logical transport that is nominally aligned with the next (odd-numbered) slot start. Such a packet may occupy up to five time slots, depending on the packet type. On a broadcast logical transport, no slaves are allowed to respond.

A special characteristic of the basic piconet physical channel is the use of some reserved slots to transmit a beacon train. The beacon train is used only if the piconet physical channel has parked slaves connected to it. In this situation, the master transmits a packet in the reserved beacon train slots. (These packets are used by the slave to resynchronize to the piconet physical channel.) The master may transmit packets from any logical transport in these slots, providing there is a transmission starting in each of the slots. In the case where there is information from the parked slave broadcast (PSB) logical transport to be transmitted, then this is transmitted in the beacon train slots and may take priority over any other logical transport.

A basic piconet physical channel may be shared by any number of devices, limited only by the resources available on the piconet master device. Only one device is the piconet master, all others being piconet slaves. All communication is between the master and slave devices. There is no direct communication between slave devices on the piconet physical channel.

There is, however, a limitation on the number of logical transports that can be supported within a piconet. This means that there is a limit to the number of these devices that can be actively involved in exchanging data with the master.

The basic piconet physical channel supports a number of physical links, logical transports, logical links, and L2CAP channels used for general purpose communications.

6.4.4.2 Adapted piconet physical channel

The adapted piconet physical channel differs from the basic piconet physical channel in two ways. First, the frequencies on which the slaves transmit are the same as the preceding master transmit frequency. In other words, the frequency is not recomputed between master and subsequent slave packets. Second, the adapted type can be based on fewer than the full 79 frequencies. A number of frequencies may be excluded from the hopping pattern by being marked as “unused.” The remainder of the 79 frequencies are included. The two sequences are the same except that, whenever the basic pseudo-random hopping sequence would have selected an unused frequency, it is replaced with an alternative chosen from the used set.

Because the adapted piconet physical channel uses the same timing and access code as the basic piconet physical channel, the two channels are often coincident. This provides a deliberate benefit as it allows slaves in either the basic or the adapted piconet physical channel to adjust their synchronization to the master.

The topology and supported layers of the adapted piconet physical channels are identical to the basic piconet physical channel.

6.4.4.3 Inquiry scan physical channel

In order for a device to be discovered, an inquiry scan physical channel is used. A discoverable device listens for inquiry requests on its inquiry scan physical channel and then sends responses to these requests. In order for a device to discover other devices, it iterates (hops) through all possible inquiry scan physical channel frequencies in a pseudo-random fashion, sending an inquiry request on each frequency and listening for any response.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Inquiry scan physical channels follow a slower hopping pattern and use an access code to distinguish between occasional occupancy of the same RF by two collocated devices using different physical channels.

The access code used on the inquiry scan physical channel is taken from a reserved set of inquiry access codes (IACs) that are shared by all devices. One access code is used for general inquiries, and a number of additional access codes are reserved for limited inquiries. Each device has access to a number of different inquiry scan physical channels. As all of these channels share an identical hopping pattern, a discoverable device may concurrently occupy more than one inquiry scan physical channel if it is capable of concurrently correlating more than one access code.

A discoverable device using one of its inquiry scan physical channel remains passive until it receives an inquiry message on this channel from another device. This is identified by the appropriate IAC. The inquiry scanning device will then follow the inquiry response procedure to return a response to the inquiring device.

In order for a device to discover other devices, it uses the inquiry scan physical channel of these devices to send inquiry requests. As it has no prior knowledge of the devices to discover, it cannot know the exact characteristics of the inquiry scan physical channel.

The device takes advantage of the fact that inquiry scan physical channels have a reduced number of hop frequencies and a slower rate of hopping. The inquiring device transmits inquiry requests on each of the inquiry scan hop frequencies and listens for an inquiry response. This is done at a faster rate, allowing the inquiring device to cover all inquiry scan frequencies in a reasonably short time period.

Inquiring and discoverable devices use a simple exchange of packets to fulfill the inquiring function. The topology formed during this transaction is a simple and transient point-to-point connection.

During the exchange of packets between an inquiring and discoverable device, it may be considered that a temporary physical link exists between these devices.

6.4.4.4 Page scan physical channel

A connectable device (i.e., one that is prepared to accept connections) does so using a page scan physical channel. A connectable device listens for page requests on its page scan physical channel and enters into a sequence of exchanges with this device. In order for a device to connect to another device, it iterates (hops) through all page scan physical channel frequencies in a pseudo-random fashion, sending a page request on each frequency and listening for any response.

The page scan physical channel uses an access code derived from the scanning device's device address to identify communications on the channel. The page scan physical channel uses a slower hopping rate than the hop rate of the basic and adapted piconet physical channels. The hop selection algorithm uses the native device clock of the scanning device as an input.

A connectable device using its page scan physical channel remains passive until it receives a page request from another device. This is identified by the page scan CAC. The two devices will then follow the page procedure to form a connection. Following a successful conclusion of the page procedure, both devices switch to the basic piconet physical channel that is characterized by having the paging device as master.

In order for a device to connect to another device, it uses the page scan physical channel of the target device to send page requests. If the paging device does not know the phase of the target device's page scan physical channel, it, therefore, does not know the current hop frequency of the target device. The paging device transmits page requests on each of the page scan hop frequencies and listens for a page response. This is done at a faster hop rate, allowing the paging device to cover all page scan frequencies in a reasonably short time period.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The paging device may have some knowledge of the target device's CLKN (indicated during a previous inquiry transaction between the two devices or as a result of a previous involvement in a piconet with the device), in which case it is able to predict the phase of the target device's page scan physical channel. It may use this information to optimize the synchronization of the paging and page scanning process and speed up the formation of the connection.

Paging and connectable devices use a simple exchange of packets to fulfill the paging function. The topology formed during this transaction is a simple and transient point-to-point connection.

During the exchange of packets between a paging and connectable device, a temporary physical link exists on the page scan physical channel between these devices.

A physical link represents a BB connection between devices. A physical link is always associated with exactly one physical channel (although a physical channel may support more than one physical link).

Within the system, a physical link is a virtual concept that has no direct representation within the structure of a transmitted packet. The Access Code Packet field, together with the clock and address of the master device are used to identify a physical channel. The physical link may be identified by association with the logical transport, as each logical transport is received only on one physical link.

Some physical link types have properties that may be modified. An example of this is the transmit power for the link. Other physical link types have no such properties. In the case of physical links with modifiable properties, the LMP is used to adapt these properties. As the LMP is supported at a higher layer (by a logical link), the appropriate physical link is identified by implication from the logical link that transports the LM signalling.

In the situation where a transmission is broadcasted over a number of different physical links, then the transmission parameters are selected to be suitable for all of the physical links.

6.4.5 Physical links

6.4.5.1 Links supported by basic and adapted piconet physical channels

The basic and adapted piconet physical channels support a physical link that may be active or parked. The physical link is a point-to-point link between the master and a slave. It is always present when the slave is synchronized in the piconet.

6.4.5.1.1 Active physical link

The physical link between a master and a slave device is active if a default ACL logical transport exists between the devices. Active physical links have no direct identification of their own, but are identified by association with the default ACL logical transport identity with which there is a one-to-one correspondence.

An active physical link has the associated properties of radio transmit power in each direction. Transmissions from slave devices are always directed over the active physical link to the master and use the transmit power that is a property of this link in the slave-to-master direction. Transmissions from the master may be directed over a single, active physical link (to a specific slave) or over a number of physical links (to a group of slaves in the piconet). In the case of point-to-point transmissions, the master uses the appropriate transmit power for the physical link in question. (In the case of point-to-multipoint transmissions, the master uses a transmit power appropriate for the set of devices addressed.)

Active physical links may be placed into HOLD or SNIFF mode. The effect of these modes is to modify the periods when the physical link is active and may carry traffic. Synchronous logical transports are not affected by these modes and continue according to their predefined scheduling behavior. The default ACL

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

logical transport and other links with undefined scheduling characteristics are subject to the mode of the active physical link.

6.4.5.1.2 Parked physical link

The physical link between a master and a slave device is parked when the slave remains synchronized in the piconet, but has no default ACL logical transport. Such a slave is also said to be parked. A beacon train is used to provide regular synchronization to all parked slaves connected to the piconet physical channel. A PSB logical transport is used to allow communication of a subset of LMP signalling and broadcast L2CAP to parked slaves. The PSB logical transport is closely associated with the beacon train.

A slave is parked (i.e., its active link is changed to a parked link) using the park procedure. The master is not allowed to park a slave that has any user-created logical transport supported by the physical link. These logical transports are first removed, and any L2CAP channels that are built on these logical transports are also removed. The broadcast logical transport and default ACL logical transports are not considered as user created and are not explicitly removed. When the active link is replaced with a parked link, the default ACL logical transport is implicitly removed. The supported logical links and L2CAP channels remain in existence, but become suspended. It is not possible to use these links and L2CAP channels to transport signalling or data while the active link is absent.

A parked slave may become active using the unpark procedure. This procedure is requested by the slave at an access window and initiated by the master. Following the unpark procedure, the parked physical link is changed to an active physical link, and the default ACL logical transport is recreated. L2CAP channels that were suspended during the most recent park procedure are associated with the new default ACL logical transport and become active again.

Parked links do not support radio power control, as there is no feedback path from parked slaves to the piconet master that can be used to signal received signal strength at the slave or for the master to measure received signal strength from the slave. Transmissions are carried out at nominal power on parked links.

Parked links use the same physical channel as their associated active link. If a master manages a piconet that contains parked slaves using the basic piconet physical channel and also parked slaves using the adapted piconet physical channel, then it must create a PSB logical transport (and associated transport) for each of these physical channels.

A parked slave may use the inactive periods of the PSB logical transport to save power, or it may carry out activities on other physical channels unrelated to the piconet within which it is parked.

6.4.5.2 Links supported by scanning physical channels

In the case of the inquiry and page scan physical channels, the physical link exists for a relatively short time and cannot be controlled or modified in any way. These types of physical link are not further elaborated.

6.4.6 Logical links and logical transports

A variety of logical links are available to support different application data transport requirements. Each logical link is associated with a logical transport, which has a number of characteristics. These characteristics include flow control, acknowledgment and repeat mechanisms, sequence numbering, and scheduling behavior. Logical transports are able to carry different types of logical links (depending on the type of the logical transport). In the case of some of the IEEE 802.15.1 logical links, these are multiplexed onto the same logical transport. Logical transports may be carried by active physical links on either the basic or the adapted piconet physical channel.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Logical transport identification and real-time (link control) signalling are carried in the packet header, and for some logical links identification is carried in the payload header. Control signalling that does not require single slot response times is carried out using the LMP.

Table 3 lists all of the logical transport types, the supported logical link types, the types of physical links and physical channels that can support them, and a brief description of the purpose of the logical transport.

Table 3—Logical transport types

Logical transport	Links supported	Supported by	Overview
Asynchronous connection-oriented (ACL[a])	Control (LMP) ACL-C User (L2CAP) ACL-U	Active physical link, basic or adapted physical channel	Reliable or time-bounded, bidirectional, point-to-point.
Synchronous connection-oriented (SCO)	Stream (unframed) SCO-S	Active physical link, basic or adapted physical channel	Bidirectional, symmetric, point-to-point, audio-visual channels. Used for 64 kb/s constant rate data.
Extended synchronous connection-oriented (eSCO)	Stream (unframed) eSCO-S	Active physical link, basic or adapted physical channel	Bidirectional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission. Used for constant rate data synchronized to the master's CLKN.
Active slave broadcast (ASB)	User (L2CAP) ASB-U	Active physical link, basic or adapted physical channel	Unreliable, unidirectional broadcast to any devices synchronized with the physical channel. Used for broadcast L2CAP groups.
Parked slave broadcast (PSB)	Control (LMP) PSB-C User (L2CAP) PSB-U	Parked physical link, basic or adapted physical channel	Unreliable, unidirectional broadcast to all piconet devices. Used for LMP and L2CAP traffic to parked devices and for access requests from parked devices.

[a]It is clear that the most obvious abbreviation for asynchronous connection-oriented logical transport is ACO. However, this acronym has an alternative meaning in this standard. To avoid confusion between two possible meanings for ACO, the decision was made to retain the ACL abbreviation for the asynchronous connection-oriented logical transport.

The names given to the logical links and logical transports reflect some of the names used in IEEE Std 802.15.1-2002 in order to provide some degree of familiarity and continuation. However, these names do not reflect a consistent scheme, which is outlined later in this clause.

The classification of each link type follows from a selection procedure within three categories: casting (see 6.4.6.1), scheduling and acknowledgment scheme (see 6.4.6.2), and class of data (see 6.4.6.3).

6.4.6.1 Casting

The first category is that of casting. This may be either unicast or broadcast. There are no multicast links defined in this standard.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— *Unicast links.* Unicast links exist between exactly two endpoints. Traffic may be sent in either direction on unicast links. All unicast links are connection-oriented; a connection procedure takes place before the link may be used. In the case of the default ACL links, the connection procedure is an implicit step within the general paging procedure used to form ad hoc piconets.
— *Broadcast links.* Broadcast links exist between one source device and one or more receiver devices. Traffic is unidirectional; it is sent only from the source devices to the receiver devices. Broadcast links are connectionless; there is no procedure to create these links, and data may be sent over them at any time. Broadcast links are unreliable, and there is no guarantee that the data will be received.

6.4.6.2 Scheduling and acknowledgment scheme

The second category relates to the scheduling and acknowledgment scheme of the link and implies the type of traffic that is supported by the link. These are synchronous, isochronous, or asynchronous. There are no specific isochronous links defined in IEEE 802.15.1, although the default ACL link can be configured to operate in this fashion.

— *Synchronous links.* Synchronous links provide a method of associating the piconet clock with the transported data. This is achieved by reserving regular slots on the physical channel and transmitting fixed-size packets at these regular intervals. Such links are suitable for constant rate isochronous data.
— *Asynchronous links.* Asynchronous links provide a method for transporting data that have no time-based characteristics. The data are normally expected to be retransmitted until successfully received, and each data entity can be processed at any time after receipt, without reference to the time of receipt of any previous or successive entity in the stream (providing the ordering of data entities is preserved).
— *Isochronous links.* Isochronous links provide a method for transporting data that have time-based characteristics. The data may be retransmitted until received or expired. The data rate on the link need not be constant (this being the main difference from synchronous links).

6.4.6.3 Class of data

The final category is related to the class of data that are carried by the link. This is either control (LMP) data or user data. The user data category is subdivided into L2CAP (or framed) data and stream (or unframed) data.

— *Control links.* Control links are used only for transporting LMP messages between two LMs. These links are invisible above the BB layer and cannot be directly instantiated, configured, or released by applications, other than by the use of the connection and disconnection services that have this effect implicitly. Control links are always multiplexed with an equivalent L2CAP link onto an ACL logical transport. Subject to the rules defining the ARQ scheme, the control link traffic always takes priority over the L2CAP link traffic.
— *L2CAP links.* L2CAP links are used to transport L2CAP PDUs, which may carry the L2CAP signalling channel (on the default ACL-U logical link only) or framed user data submitted to user-instantiated L2CAP channels. L2CAP frames submitted to the BB may be larger than the available BB packets. A LCP embedded within the LLID field preserves the frame-start and frame-continuation semantics when the frame is transmitted in a number of fragments to the receiver.
— *Stream links.* Stream links are used to transport user data with no inherent framing that should be preserved when delivering the data. Lost data may be replaced by padding at the receiver.

6.4.6.4 Asynchronous connection-oriented (ACL)

The ACL logical transport is used to carry LMP and L2CAP control signalling and best effort asynchronous user data. The ACL logical transport uses a simple 1-bit acknowledgment scheme to provide simple channel

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

reliability. Every active slave device within a piconet has one ACL logical transport to the piconet master, known as the default ACL.

The default ACL is created between the master and the slave when a device joins a piconet (i.e., connects to the basic piconet physical channel). This default ACL is assigned an LT_ADDR by the piconet master. This LT_ADDR is also used to identify the active physical link when required (or as a piconet active member identifier, effectively for the same purpose).

The LT_ADDR for the default ACL is reused for synchronous connection-oriented (SCO) logical transports between the same master and slave. (This is for reasons of compatibility with earlier standards.) Thus the LT_ADDR is not sufficient on its own to identify the default ACL. However, the packet types used on the ACL are different from those used on the SCO logical transport. Therefore, the ACL logical transport can be identified by the LT_ADDR field in the packet header in combination with the Packet Type field.

The default ACL may be used for isochronous data transport by configuring it to automatically flush packets after the packets have expired.

If the default ACL is removed from the active physical link, then all other logical transports that exist between the master and the slave are also removed. In the case of unexpected loss of synchronization to the piconet physical channel, the physical link and all logical transports and logical links cease to exist at the time that this synchronization loss is detected.

A device may remove its default ACL (and by implication its active physical link), but remain synchronized to the piconet. This procedure is known as *parking*; and a device that is synchronized to the piconet, but has no active physical link, is parked within that piconet.

When the device transitions to the PARK state, the default ACL logical links that are transported on the default ACL logical transport remain in existence, but become suspended. No data may be transferred across a suspended logical link. When the device transitions from the PARK state back into ACTIVE state, a new default ACL logical transport is created (it may have a different LT_ADDR from the previous one); and the suspended logical links are attached to this default ACL and become active once again.

6.4.6.5 Synchronous connection-oriented (SCO)

The SCO logical transport is a symmetric, point-to-point channel between the master and a specific slave. The SCO logical transport reserves slots on the physical channel and can, therefore, be considered as a circuit-switched connection between the master and the slave. SCO logical transports carry 64 kb/s of information synchronized with the piconet clock. Typically this information is an encoded voice stream. Three different SCO configurations exist, offering a balance between robustness, delay, and bandwidth consumption.

Each SCO-S logical link is supported by a single SCO logical transport, which is assigned the same LT_ADDR as the default ACL logical transport between the devices. Therefore, the LT_ADDR field is not sufficient to identify the destination of a received packet. Because the SCO links use reserved slots, a device uses a combination of the LT_ADDR, the slot numbers (a property of the physical channel), and the packet type to identify transmissions on the SCO link.

The reuse of the default ACL's LT_ADDR for SCO logical transports is due to legacy behavior from IEEE Std 802.15.1-2002. In this earlier version of Bluetooth, the LT_ADDR (then known as the *active member address*) was used to identify the piconet member associated with each transmission. This was not easily extensible for enabling more logical links; therefore, the purpose of this field was redefined for the new features. Some of IEEE Std 802.15.1-2002 features, however, do not cleanly fit into the more formally described architecture.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Although slots are reserved for the SCO, it is permissible to use a reserved slot for traffic from another channel that has a higher priority. This may be required as a result of QoS commitments or to send LMP signalling on the default ACL when the physical channel bandwidth is fully occupied by SCOs. As SCOs carry different packet types than ACLs do, the packet type is used to identify SCO traffic (in addition to the slot number and LT_ADDR).

There are no further architectural layers defined by the core specification that are transported over an SCO link. A number of standard formats are defined for the 64 kb/s stream that is transported, or an unformatted stream is allowed where the application is responsible for interpreting the encoding of the stream.

6.4.6.6 Extended synchronous connection-oriented (eSCO)

The eSCO logical transport is a symmetric or asymmetric, point-to-point link between the master and a specific slave. The eSCO reserves slots on the physical channel and can, therefore, be considered as a circuit-switched connection between the master and the slave. eSCO links offer a number of extensions over the standard SCO links, in that they support a more flexible combination of packet types and selectable data contents in the packets and selectable slot periods, allowing a range of synchronous bit rates to be supported.

eSCO links also can offer limited retransmission of packets (unlike SCO links where there is no retransmission). If these retransmissions are required, they take place in the slots that follow the reserved slots; otherwise, the slots may be used for other traffic.

Each eSCO-S logical link is supported by a single eSCO logical transport, identified by an LT_ADDR that is unique within the piconet for the duration of the eSCO. eSCO-S links are created using LM signalling and follow scheduling rules similar to SCO-S links.

There are no further architectural layers defined by the core specification that are transported over an eSCO-S link. Instead, applications may use the data stream for whatever purpose they require, subject to the transport characteristics of the stream being suitable for the data being transported.

6.4.6.7 Active slave broadcast (ASB)

The ASB logical transport is used to transport L2CAP user traffic to all devices in the piconet that are currently connected to the physical channel that is used by the ASB. There is no acknowledgment protocol, and the traffic is unidirectional from the piconet master to the slaves. The ASB channel may be used for L2CAP group traffic (a legacy of IEEE Std 802.15.1-2002 and the Bluetooth 1.1 specification) and is never used for L2CAP connection-oriented channels, L2CAP control signalling, or LMP control signalling.

The ASB logical transport is inherently unreliable because of the lack of acknowledgment. To improve the reliability, each packet is transmitted a number of times. An identical sequence number (SEQN) is used to assist with filtering retransmissions at the slave device.

The ASB logical transport is identified by a reserved LT_ADDR. (The reserved LT_ADDR address is also used by the PSB logical transport.) An active slave will receive traffic on both logical transports and cannot readily distinguish between them. As the ASB logical transport does not carry LMP traffic, an active slave can ignore packets received over the LMP logical link on the ASB logical transport. However, L2CAP traffic transmitted over the PSB logical transport is also received by active slaves on the ASB logical transport and cannot be distinguished from L2CAP traffic sent on the ASB transport.

An ASB is implicitly created whenever a piconet exists, and there is always one ASB associated with each of the basic and adapted piconet physical channels that exist within the piconet. Because the basic and adapted piconet physical channels are mostly coincident, a slave device cannot distinguish which of the ASB channels is being used to transmit the packets. This adds to the general unreliability of the ASB channel (although it is, perhaps, no more unreliable than general missed packets).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A master device may decide to use only one of its two possible ASBs (when it has both a basic and adapted piconet physical channel), as with sufficient retransmissions it is possible to address both groups of slaves on the same ASB channel.

The ASB channel is never used to carry LMP or L2CAP control signals.

6.4.6.8 Parked slave broadcast (PSB)

The PSB logical transport is used for communications between the master and slaves that are parked (i.e., have given up their default ACL logical transport). The PSB link is the only logical transport that exists between the piconet master and parked slaves.

The PSB logical transport is more complex than the other logical transports as it consists of a number of phases, each having a different purpose. These phases are the control information phase (used to carry the LMP logical link), the user information phase (used to carry the L2CAP logical link), and the access phase (used to carry BB signalling). The control information and broadcast information phases are usually mutually exclusive as only one of them can be supported in a single beacon interval. (Even if there is no control or user information phase, the master is still required to transmit a packet in the beacon slots so that the parked slaves can resynchronize.) The access phase is normally present unless cancelled in a control information message.

The control information phase is used for the master to send information to the parked slaves containing modifications to the PSB transport attributes, modifications to the beacon train attributes, or a request for a parked slave to become active in the piconet (known as *unparking*). This control information is carried in LMP messages on the LMP logical link. (The control information phase is also present in the case of a user information phase where the user information requires more than one BB packet.)

Packets in the control information phase are always transmitted in the physical channel beacon train slots and cannot be transmitted on any other slots. The control information occupies a single **DM1** packet and is repeated in every beacon train slot within a single beacon interval. (If there is no control information, then there may be a user information phase that uses the beacon slots. If neither phase is used, then the beacon slots are used for other logical transport traffic or for NULL packets.)

The user information phase is used for the master to send L2CAP packets that are destined for all piconet slaves. User information may occupy one or more BB packets. If the user information occupies a single packet, then the user information packet is repeated in each of the piconet physical channel beacon train slots.

If the user information occupies more than one BB packet, then it is transmitted in slots after the beacon train (the broadcast scan window), and the beacon slots are used to transmit a control information phase message that contains the timing attributes of this broadcast scan window. This is required so that the parked slaves remain connected to the piconet physical channel to receive the user information.

The access phase is normally present unless temporarily cancelled by a control message carried in the control information broadcast phase. The access window consists of a sequence of slots that follow the beacon train. In order for a parked slave to become active in the piconet, it may send such an access request to the piconet master during the access window. Each parked slave is allocated an access request address (AR_ADDR) (not necessarily unique) that controls when, during the access window, the slave requests access.

The PSB logical transport is identified by the reserved LT_ADDR of 0. This reserved LT_ADDR is also used by the ASB logical transport. Parked slaves are not normally confused by the duplicated use of the LT_ADDR as they are connected to the piconet physical channel only during the time that the PSB transport is being used.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

6.4.6.9 Logical links

Some logical transports are capable of supporting different logical links, either concurrently multiplexed, or one of the choice. Within such logical transports, the logical link is identified by the LLID bits in the payload header of BB packets that carry a data payload. The logical links distinguish between a limited set of core protocols that are able to transmit and receive data on the logical transports. Not all of the logical transports are able to carry all of the logical links (the supported mapping is shown in Figure 3). In particular, the SCO and eSCO logical transports are able to carry only constant data rate streams, and these are uniquely identified by the LT_ADDR. Such logical transports use only packets that do not contain a payload header, as their length is known in advance, and no LLID is necessary.

6.4.6.10 ACL control logical link (ACL-C)

The ACL-C logical link is used to carry LMP signalling between devices in the piconet. The control link is carried only on the default ACL logical transport and on the PSB logical transport (in the control information phase). The ACL-C link is always given priority over the ACL-U link when carried on the same logical transport.

6.4.6.11 Asynchronous/isochronous user logical link (ACL-U)

The ACL-U logical link is used to carry all asynchronous and isochronous framed user data. The ACL-U link is carried on all but the synchronous logical transports. Packets on the ACL-U link are identified by one of two reserved LLID values. One value is used to indicate whether the BB packet contains the start of an L2CAP frame and the other indicates a continuation of a previous frame. This ensures correct synchronization of the L2CAP reassembler following flushed packets. The use of this technique removes the need for a more complex L2CAP header in every BB packet (the header is required only in the L2CAP start packets), but adds the requirement that a complete L2CAP frame shall be transmitted before a new one is transmitted. (An exception to this rule being the ability to flush a partially transmitted L2CAP frame in favor of another L2CAP frame.)

6.4.6.12 SCO/eSCO streaming logical links (SCO-S/eSCO-S)

SCO-S and eSCO-S logical links are used to support isochronous data delivered in a stream without framing. These links are associated with a single logical transport, where data are delivered in constant-sized units at a constant rate. There is no LLID within the packets on these transports, as only a single logical link can be supported, and the packet length and scheduling period are predefined and remain fixed during the lifetime of the link.

Variable rate isochronous data cannot be carried by the SCO-S or eSCO-S logical links. In this case, the data must be carried on ACL-U logical links, which use packets with a payload header. All versions to date of this standard have some limitations when supporting variable-rate isochronous data concurrently with reliable user data.

6.5 L2CAP channels

The L2CAP provides a multiplexing role allowing many different applications to share the resources of an ACL-U logical link between two devices. Applications and service protocols interface with the L2CAP using a channel-oriented interface to create connections to equivalent entities on other devices.

L2CAP channel endpoints are identified to their clients by a channel identifier (CID). This is assigned by the L2CAP, and each L2CAP channel endpoint on any device has a different CID.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

L2CAP channels may be configured to provide an appropriate QoS to the application. The L2CAP maps the channel onto the ACL-U logical link.

L2CAP supports channels that are connection-oriented and others that are group-oriented. Group-oriented channels may be mapped onto the ASB-U logical link or implemented as iterated transmission to each member in turn over an ACL-U logical link.

Apart from the creation, configuration, and dismantling of channels, the main role of L2CAP is to multiplex SDUs from the channel clients onto the ACL-U logical links and to carry out a simple level of scheduling, selecting SDUs according to relative priority.

L2CAP can provide a per-channel flow control with the peer L2CAPs. This option is selected by the application when the channel is established. L2CAP can also provide enhanced error detection and retransmission to

— Reduce the probability of undetected errors being passed to the application
— Recover from loss of portions of the user data when the BB protocol performs a flush on the ACL-U logical link

In the case where an HCI is present, the L2CAP is also required to segment L2CAP SDUs into fragments that will fit into the BB buffers and also to operate a token-based flow control procedure over the HCI, submitting fragments to the BB only when allowed to do so. This may affect the scheduling algorithm.

6.6 Communication topology

Any time an IEEE 802.15.1-2005 link is formed, it is within the context of a piconet. A piconet consists of two or more devices that occupy the same physical channel (in other words, they are synchronized to a common clock and hopping sequence). The common (piconet) clock is identical to the CLKN of one of the devices in the piconet, known as the master of the piconet; and the hopping sequence is derived from the master's CLKN and the master's device address. All other synchronized devices are referred to as slaves in the piconet. The terms *master* and *slave* are used only when describing these roles in a piconet.

6.6.1 Piconet topology

Within a common location, a number of independent piconets may exist. Each piconet has a different physical channel (i.e., a different master device and an independent piconet clock and hopping sequence).

A device may participate concurrently in two or more piconets. It does this on a TDM basis. A device can never be a master of more than one piconet. (Since the piconet is defined by synchronization to the master's CLKN, it is impossible to be the master of two or more piconets.) A device may be a slave in many independent piconets.

A device that is a member of two or more piconets is said to be involved in a scatternet. Involvement in a scatternet does not necessarily imply any network routing capability or function in the device. The core protocols do not, and are not intended to, offer such functionality, which is the responsibility of higher level protocols and is outside the scope of this standard.

In Figure 6, an example of topology is shown that demonstrates a number of the architectural features described below. Device A is a master in a piconet (represented by the shaded area and known as piconet A) with devices B, C, D and E as slaves. Two other piconets are shown:

— With device F as master (known as piconet F) and devices E, G, and H as slaves
— With device D as master (known as piconet D) and device J as slave

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

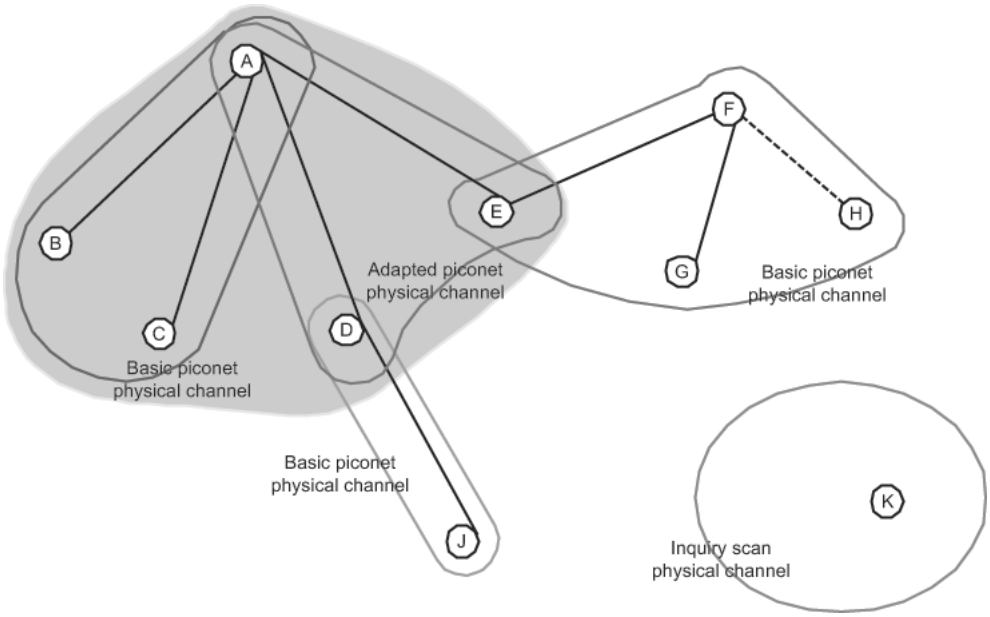


Figure 6—Example of IEEE 802.15.1-2005 topology

In piconet A, there are two physical channels. Devices B and C are using the basic piconet physical channel as they do not support AFH. Devices D and E are capable of supporting AFH and are using the adapted piconet physical channel. Device A is capable of AFH and operates in a TDM basis on both physical channels according to which slave is being addressed.

Piconets D and F are both using only a basic piconet physical channel. In the case of piconet D, this is because device J does not support the adaptive hopping mode. Although device D supports adaptive hopping, it cannot use it in this piconet. In piconet F, device F does not support adaptive hopping; therefore, it cannot be used in this piconet.

Device K is shown in the same locality as the other devices. It is not currently a member of a piconet, but has services that it offers to other devices. It is currently listening on its inquiry scan physical channel, awaiting an inquiry request from another device.

Physical links (one per slave device) are represented in the figure by lines connecting the devices. The solid lines represent an active physical link, and the dashed line represents a parked physical link. Device H is parked; hence, the physical link between the master (device F) and the slave (device H) is shown as parked.

Logical transports, logical links, and L2CAP channels are used to provide capabilities for the transport of data, but are not shown on this figure. They are described in more detail in 6.4.6 and 6.5.

6.6.2 Operational procedures and modes

The typical operational mode of a device is to be connected to other devices (in a piconet), which exchange data with that device. Since IEEE 802.15.1 is an ad hoc wireless communications technology, there are also a number of operational procedures that enable piconets to be formed so that the subsequent communications can take place. Procedures and modes are applied at different layers in the architecture; therefore, a device may be engaged in a number of these procedures and modes concurrently.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

6.6.2.1 Inquiry (discovering) procedure

All IEEE 802.15.1 devices use the inquiry procedure to discover nearby devices or to be discovered by devices in their locality.

The inquiry procedure is asymmetrical. A device that tries to find other nearby devices is known as an *inquiring device* and actively sends inquiry requests. Devices that are available to be found are known as *discoverable devices*; they listen for these inquiry requests and send responses. The inquiry procedure uses a special physical channel for the inquiry requests and responses.

Both inquiring and discoverable devices may already be connected to other devices in a piconet. Any time spent inquiring or occupying the inquiry scan physical channel needs to be balanced with the demands of the QoS commitments on existing logical transports.

The inquiry procedure does not make use of any of the architectural layers above the physical channel, although a transient physical link may be considered to be present during the exchange of inquiry and inquiry response information.

6.6.2.2 Paging (connecting) procedure

The procedure for forming connections is asymmetrical and requires that one device (i.e., the *paging device*) carry out the page (connection) procedure while the other device (i.e., the *connectable device*) is connectable (page scanning). The procedure is targeted so that the page procedure is responded to only by one specified device.

The connectable device uses a special physical channel to listen for connection request packets from the paging (or connecting) device. This physical channel has attributes that are specific to the connectable device; hence, only a paging device with knowledge of the connectable device is able to communicate on this channel.

Both paging and connectable devices may already be connected to other devices in a piconet. Any time spent paging or occupying the page scan physical channel needs to be balanced with the demands of the QoS commitments on existing logical transports.

6.6.2.3 Connected mode

After a successful connection procedure, the devices are physically connected to each other within a piconet. This means that there is a piconet physical channel to which they are both connected, there is a physical link between the devices, and there are default ACL-C and ACL-U logical links. When in the connected mode, it is possible to create and release additional logical links and to change the modes of the physical and logical links while remaining connected to the piconet physical channel. It is also possible for the device to carry out inquiry, paging, or scanning procedures or to be connected to other piconets without needing to disconnect from the original piconet physical channel.

Additional logical links are created using the LM that exchanges LMP messages with the remote device to negotiate the creation and settings for these links. Default ACL-C and ACL-U logical links are always created during the connection process, and these are used for LMP messages and the L2CAP signalling channel, respectively.

It is noted that two default logical links are created when two units are initially connected. One of these links (ACL-C) transports the LMP control protocol and is invisible to the layers above the LM. The other link (ACL-U) transports the L2CAP signalling protocol and any multiplexed L2CAP best-effort channels. It is common to refer to a default ACL logical transport, which can be resolved by context, but typically refers to the default ACL-U logical link. Also note that these two logical links share a logical transport.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

During the time that a slave device is actively connected to a piconet, there is always a default ACL logical transport between the slave and the master device. There are two methods of deleting the default ACL logical transport.

The first method is to detach the device from the piconet physical channel, at which time the entire hierarchy of L2CAP channels, logical links, and logical transports between the devices is deleted.

The second method is to place the physical link to the slave device in the PARK state, at which time it gives up its default ACL logical transport. This is allowed only if all other logical transports have been deleted (except for the ASB logical transport that cannot be explicitly created or deleted). It is not allowed to park a device while it has any logical transports other than the default ACL and ASB logical transports.

When the slave device physical link is parked, its default ACL logical transport is released and the ASB logical transport is replaced with a PSB logical transport. The ACL-C and ACL-U logical links that are multiplexed onto the default ACL logical transport remain in existence, but cannot be used for the transport of data. The LM on the master device restricts itself to the use of LMP messages that are allowed to be transported over the parked slave broadcast control (PSB-C) logical link. The channel manager and L2CAP resource manager ensure that no L2CAP unicast data traffic is submitted to the controller while the device is parked. The channel manager may decide to manage the parking and unparking of the device as necessary to allow data to be transported.

6.6.2.4 HOLD mode

HOLD mode is not a general device mode, but applies to unreserved slots on the physical link. When in this mode, the physical link is active only during slots that are reserved for the operation of the synchronous link types SCO and eSCO. All asynchronous links are inactive. HOLD modes operate once for each invocation and are then exited when complete, returning to the previous mode.

6.6.2.5 SNIFF mode

SNIFF mode is not a general device mode, but applies to the default ACL logical transports. When in this mode, the availability of these logical transports is modified by defining a duty cycle consisting of periods of presence and absence. Devices that have their default ACL logical transports in SNIFF mode may use the absent periods to engage in activity on another physical channel or to enter reduced power mode. SNIFF mode affects only the default ACL logical transports (i.e., their shared ACL logical transport) and does not apply to any additional SCO or eSCO logical transports that may be active. The periods of presence and absence of the physical link on the piconet physical channel is derived as a union of all logical transports that are built on the physical link.

Note that broadcast logical transports have no defined expectations for presence or absence. A master device should aim to schedule broadcasts to coincide with periods of physical link presence within the piconet physical channel, but this may not always be possible or practical. Repetition of broadcasts is defined to improve the possibilities for reaching multiple slaves without overlapping presence periods. However, broadcast logical transports cannot be considered to be reliable.

6.6.2.6 PARK state

A slave device may remain connected to a piconet, but have its physical link in PARK state. In this state, the device cannot support any logical links to the master with the exception of the PSB-C and PSB-U logical links that are used for all communication between the piconet master and the parked slave.

When the physical link to a slave device is parked, this means that there are restrictions on when the master and slave may communicate, defined by the PSB logical transport parameters. During times when the PSB

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

logical transport is inactive (or absent), then the devices may engage in activity on other physical channels or enter reduced power mode.

6.6.2.7 Role switch procedure

The role switch procedure is a method for swapping the roles of two devices connected in a piconet. The procedure involves moving from the physical channel that is defined by the original master device to the physical channel that is defined by the new master device. In the process of swapping from one physical channel to the next, the hierarchy of physical links and logical transports are removed and rebuilt, with the exception of the ASB and PSB logical transports that are implied by the topology and are not preserved. After the role switch, the original piconet physical channel may cease to exist or may be continued if the original master had other slaves that are still connected to the channel.

The procedure copies only the default ACL logical links and supporting layers to the new physical channel. Any additional logical transports are not copied by this procedure; and if required, this must be carried out by higher layers. The LT_ADDRs of any affected transports may not be preserved as the values may already be in use on the new physical channel.

If there are any QoS commitments or modes such as SNIFF mode on the original logical transports, then these are not preserved after a role switch. These must be renegotiated after the role switch has completed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

7. Physical layer (PHY)

7.1 Scope

IEEE 802.15.1-2005 devices operate in the unlicensed 2.4 GHz ISM band. A frequency hop transceiver is applied to combat interference and fading. A shaped, binary FM is applied to minimize transceiver complexity. The symbol rate is 1 Msymbol/s. For full duplex transmission, a TDD scheme is used. This clause defines the requirements for an IEEE 802.15.1-2005 radio.

Requirements are defined for two reasons:

— To provide compatibility between radios used in the system
— To define the quality of the system

The radio shall fulfill the stated requirements under the operating conditions specified in 7.5, 7.6, and 7.7.

7.1.1 Regional authorities

This standard is based on the established regulations for Europe, Japan, and North America. The standard documents listed in 7.1.1.1, 7.1.1.2, and 7.1.1.3 are only for information and are subject to change or revision at any time.

7.1.1.1 Europe

Approval Standards: European Telecommunications Standards Institute (ETSI)
Documents: EN 300 328, ETS 300-826
Approval Authority: National Type Approval Authorities

7.1.1.2 Japan

Approval Standards: Association of Radio Industries and Businesses (ARIB)
Documents: ARIB STD-T66
Approval Authority: Ministry of Post and Telecommunications (MPT)

7.1.1.3 North America

Approval Standards: Federal Communications Commission (FCC), USA
Documents: CFR47, Part 15, Sections 15.205, 15.209, 15.247, and 15.249

Approval Standards: Industry Canada (IC), Canada
Documents: GL36

Approval Authority: FCC (USA), IC (Canada)

7.1.2 Frequency bands and channel arrangement

IEEE 802.15.1 operates in the 2.4 GHz ISM band. This frequency band is from 2400 MHz to 2483.5 MHz.

RF channels are spaced 1 MHz and are ordered in channel number k as shown in Table 4. In order to comply with out-of-band regulations in each country, a guard band is used at the lower and upper band edge. See Table 5. Because the RF channels are 1 MHz wide, centered on the RF channels defined in Table 4, the operating range of IEEE 802.15.1 radios is from 2401.5 MHz to 2480.5 MHz.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 4—Operating frequency bands

Regulatory range	RF channels
2.400–2.4835 GHz	f = 2402 + k MHz, k = 0,…,78

Table 5—Guard bands

Lower guard band	Upper guard band
1.5 MHz	3 MHz

7.2 Transmitter characteristics

The requirements stated in this subclause are given as power levels at the antenna connector of the device. If the device does not have a connector, a reference antenna with 0 dBi gain is assumed.

Due to difficulty in measurement accuracy in radiated measurements, systems with an integral antenna should provide a temporary antenna connector during type approval.

If transmitting antennas of directional gain greater than 0 dBi are used, the applicable paragraphs in EN 300 328, EN 301 489-17 and FCC Part 15 shall be compensated for. The device is classified into three power classes. In Table 6, P_{min} is the minimum output power at maximum power setting.

Table 6—Power classes

Power class	Maximum output power (P_{max})	Nominal output power	Minimum output power (P_{min})	Power control
1	100 mW (20 dBm)	N/A	1 mW (0 dBm)	P_{min}<+4 dBm to P_{max} Optional: P_{min}[a] to P_{max}
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (–6 dBm)	Optional: P_{min}[a] to P_{max}
3	1 mW (0 dBm)	N/A	N/A	Optional: P_{min}[a] to P_{max}

[a]The lower power limit P_{min}<–30dBm is suggested, but is not mandatory, and may be chosen according to application needs.

Power class 1 device shall implement power control. The power control is used for limiting the transmitted power over +4 dBm. Power control capability under +4 dBm is optional and could be used for optimizing the power consumption and overall interference level. The power steps shall form a monotonic sequence, with a maximum step size of 8 dB and a minimum step size of 2 dB. A class 1 device with a maximum transmit power of +20 dBm shall be able to control its transmit power down to 4 dBm or less.

Devices with power control capability optimize the output power in a physical link with LMP commands (see Clause 9). This is done by each device checking its received signal strength indication (RSSI) and reporting to the peer if the power is above or below a desired golden range. On receiving such a report, a device that implements power control shall adjust its transmit power unless such an adjustment would take

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

the power above the device's maximum power or the maximum power for the device's power class or take the power below the device's minimum power.

In a connection, the output power shall not exceed the maximum output power of power class 2 for transmitting packets if the receiving device does not support the necessary messaging for sending the power control messages (see 9.3.1.3). In this case, the transmitting device shall comply with the rules of a class 2 or class 3 device.

If a class 1 device is paging or inquiring very close to another device, the input power can be larger than the requirement in 7.4.5. This can cause the receiving device to fail to respond. It may, therefore, be useful to page at power class 2 or 3 in addition to paging at power class 1.

7.3 Modulation characteristics

The modulation is Gaussian frequency shift keying (GFSK) (see Figure 7) with a bandwidth-bit period product, known as bandwidth time (BT), of 0.5. The modulation index shall be between 0.28 and 0.35. A binary one shall be represented by a positive frequency deviation, and a binary zero shall be represented by a negative frequency deviation. The symbol timing shall be less than ± 20 ppm.

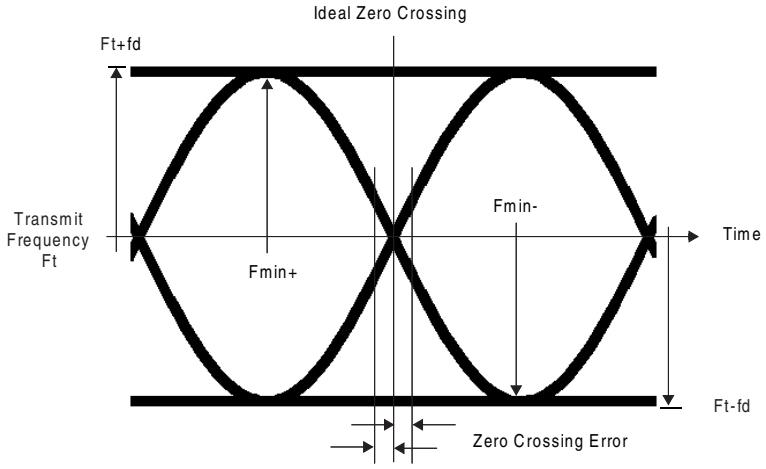


Figure 7—GFSK parameters definition

For each transmission, the minimum frequency deviation, $F_{min} = \min\{|F_{min}+|, F_{min}-\}$, which corresponds to 1010 sequence, shall be no smaller than ± 80% of the frequency deviation fd with respect to the transmit frequency Ft, which corresponds to a 00001111 sequence.

In addition, the minimum frequency deviation shall never be smaller than 115 kHz. The data transmitted have a symbol rate of 1 Msymbol/s.

The zero crossing error is the time difference between the ideal symbol period and the measured crossing time. This shall be less than ± 1/8 of a symbol period.

7.3.1 Spurious emissions

In-band spurious emissions shall be measured with a frequency hopping radio transmitting on one RF channel and receiving on a second RF channel; this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same transmit RF channel. There will be no reference

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

in this standard to out-of-ISM-band spurious emissions; the equipment manufacturer is responsible for compliance in the intended country of use.

7.3.1.1 In-band spurious emission

Within the ISM band, the transmitter shall pass a spectrum mask, given in Table 7.[8] The spectrum shall comply with the 20 dB bandwidth definition in FCC Part 15.247 and shall be measured accordingly. In addition to the FCC requirement, an adjacent channel power on adjacent channels with a difference in RF channel number of two or greater is defined. This adjacent channel power is defined as the sum of the measured power in a 1 MHz RF channel. The transmitted power shall be measured in a 100 kHz bandwidth using maximum hold. The device shall transmit on RF channel M, and the adjacent channel power shall be measured on RF channel number N. The transmitter shall transmit a pseudo-random data pattern in the payload throughout the test.

Table 7—Transmit spectrum mask

Frequency offset	Transmit power
± 500 kHz	–20 dBc
2MHz (|M – N| = 2)	–20 dBm
3MHz or greater (|M – N| ≥ 3)	–40 dBm
NOTE—If the output power is less than 0 dBm, then, wherever appropriate, the FCC's 20 dB relative requirement overrules the absolute adjacent channel power requirement stated in this table.	

Exceptions are allowed in up to three bands of 1 MHz width centered on a frequency that is an integer multiple of 1 MHz. They shall comply with an absolute value of –20 dBm.

7.3.1.2 RF tolerance

The transmitted initial center frequency shall be within ± 75 kHz from center frequency. The initial frequency accuracy is defined as being the frequency accuracy before any packet information is transmitted. Note that the frequency drift requirement is not included in the ± 75 kHz.

The limits on the transmitter center frequency drift within a packet are specified in Table 8. The different packets are defined in Clause 8.

Table 8—Maximum allowable frequency drifts in a packet

Duration of packet	Frequency drift
Maximum length 1-slot packet	± 25 kHz
Maximum length 3-slot packet	± 40 kHz
Maximum length 5-slot packet	± 40 kHz
Maximum drift rate[a]	400 Hz/μs

[a]The maximum drift rate is allowed anywhere in a packet.

[8]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement this standard.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

7.4 Receiver characteristics

The reference sensitivity level is –70 dBm.

7.4.1 Actual sensitivity level

The actual sensitivity level is defined as the input level for which a raw bit error rate (BER) of 0.1% is met. The receiver sensitivity shall be below or equal to –70 dBm with any transmitter compliant to the transmitter specified in 7.2.

7.4.2 Interference performance

The interference performance on co-channel and adjacent 1 MHz and 2 MHz shall be measured with the wanted signal 10 dB over the reference sensitivity level. For interference performance on all other RF channels, the wanted signal shall be 3 dB over the reference sensitivity level. If the frequency of an interfering signal is outside of the 2400–2483.5 MHz band, the out-of-band blocking specification (see 7.4.3) shall apply. The interfering signal shall be modulated as specified in 7.4.7. The BER shall be ≤0.1% for all the signal-to-interference ratios listed in Table 9.

Table 9—Interference performance

Frequency of interference	Ratio
Co-channel interference, $C/I_{co\text{-}channel}$	11 dB
Adjacent (1 MHz) interference, C/I_{1MHz}	0 dB
Adjacent (2 MHz) interference, C/I_{2MHz}	–30 dB
Adjacent (≥3 MHz) interference, $C/I_{\geq 3MHz}$	–40 dB
Image frequency interference[a, b], C/I_{Image}	–9 dB
Adjacent (1 MHz) interference to in-band image frequency, $C/I_{Image\pm 1MHz}$	–20 dB

[a]In-band image frequency.

[b]If the image frequency ≠ n*1 MHz, then the image reference frequency is defined as the closest n*1 MHz frequency.

If two adjacent channel specifications from Table 9 are applicable to the same channel, the more relaxed specification applies.

These specifications are to be tested only at nominal temperature conditions with a device receiving on one RF channel and transmitting on a second RF channel; this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same receive RF channel.

RF channels where the requirements are not met are called *spurious response RF channels*. Five spurious response RF channels are allowed at RF channels with a distance of ≥2 MHz from the wanted signal. On these spurious response RF channels, a relaxed interference requirement C/I = –17 dB shall be met.

7.4.3 Out-of-band blocking

The out-of-band suppression (or rejection) shall be measured with the wanted signal 3 dB over the reference sensitivity level. The interfering signal shall be a continuous wave signal. The BER shall be ≤0.1%. The out-of-band blocking shall fulfill the requirements in Table 10.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 10—Out-of-band suppression (or rejection) requirements

Interfering signal frequency	Interfering signal power level
30 – 2000 MHz	–10 dBm
2000 – 2399 MHz	–27 dBm
2484 – 3000 MHz	–27 dBm
3000 MHz – 12.75 GHz	–10 dBm

Twenty-four exceptions are permitted, which are dependent upon the given RF channel and are centered at a frequency that is an integer multiple of 1 MHz. For at least 19 of these spurious response frequencies, a reduced interference level of at least –50 dBm is allowed in order to achieve the required BER = 0.1% performance whereas, for a maximum of five of the spurious frequencies, the interference level may be assumed arbitrarily lower.

7.4.4 Intermodulation characteristics

The reference sensitivity performance, BER = 0.1%, shall be met under the following conditions:

- The wanted signal shall be at frequency f_0 with a power level 6 dB over the reference sensitivity level.
- A static sine wave signal shall be at frequency f_1 with a power level of –39 dBm.
- A modulated signal (see 7.4.7) shall be at frequency f_2 with a power level of –39 dBm.

Frequencies f_0, f_1, and f_2 shall be chosen so that $f_0 = 2f_1 - f_2$ and $|f_2 - f_1| = n*1$ MHz, where n can be 3, 4, or 5. The system shall fulfill at least one of the three alternatives (n = 3, 4, or 5).

7.4.5 Maximum usable level

The maximum usable input level at which the receiver operates shall be greater than –20 dBm. The BER shall be less than or equal to 0.1% at –20 dBm input power.

7.4.6 Receiver signal strength indicator

If a device supports RSSI, the accuracy shall be ± 6 dBm.

7.4.7 Reference signal definition

A modulated interfering signal shall be defined as follows:

Modulation = GFSK
Modulation index = 0.32 ± 1%
BT = 0.5 ± 1%
Bit rate = 1 Mbps ± 1 ppm
Modulating data for wanted signal = pseudo-random bit sequence 9 (PRBS9)
Modulating data for interfering signal = pseudo-random bit sequence 15 (PRBS15)
Frequency accuracy better than ± 1 ppm

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

7.5 Nominal test conditions

7.5.1 Nominal temperature

The nominal temperature conditions for tests shall be +15 to +35 ^{o}C. When it is impractical to carry out the test under this condition, a note to this effect, stating the ambient temperature, shall be recorded. The actual value during the test shall be recorded in the test report.

7.5.2 Nominal power source

7.5.2.1 Mains voltage

The nominal test voltage for equipment to be connected to the mains shall be the nominal mains voltage. The nominal voltage shall be the declared voltage or any of the declared voltages for which the equipment was designed. The frequency of the test power source corresponding to the ac mains shall be within 2% of the nominal frequency.

7.5.2.2 Lead-acid battery power sources used in vehicles

When radio equipment is intended for operation from the alternator-fed lead-acid battery power sources, which are standard in vehicles, then the nominal test voltage shall be 1.1 times the nominal voltage of the battery (e.g., 6 V, 12 V).

7.5.2.3 Other power sources

For operation from other power sources or types of battery (primary or secondary), the nominal test voltage shall be as declared by the equipment manufacturer. This shall be recorded in the test report.

7.6 Extreme test conditions

7.6.1 Extreme temperatures

The extreme temperature range shall be the largest temperature range given by the combination of the following:

— The minimum temperature range 0 °C to +35 °C
— The product operating temperature range declared by the manufacturer

This extreme temperature range and the declared operating temperature range shall be recorded in the test report.

7.6.2 Extreme power source voltages

Tests at extreme power source voltages specified in 7.6.2.1 through 7.6.2.4 are not required when the equipment under test is designed for operation as part of, and powered by, another system or piece of equipment. Where this is the case, the limit values of the host system or host equipment shall apply. The appropriate limit values shall be declared by the manufacturer and recorded in the test report.

7.6.2.1 Mains voltage

The extreme test voltage for equipment to be connected to an ac mains source shall be the nominal mains voltage ± 10%.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

7.6.2.2 Lead-acid battery power source used on vehicles

When radio equipment is intended for operation from the alternator-fed lead-acid battery power sources, which are standard in vehicles, then extreme test voltage shall be 1.3 and 0.9 times the nominal voltage of the battery (e.g., 6 V, 12 V).

7.6.2.3 Power sources using other types of batteries

The lower extreme test voltage for equipment with power sources using the indicated types of battery shall be as follows:

a) For Leclanché, alkaline, or lithium batteries: 0.85 times the nominal voltage of the battery
b) For mercury or nickel-cadmium batteries: 0.9 times the nominal voltage of the battery

In both cases, the upper extreme test voltage shall be 1.15 times the nominal voltage of the battery.

7.6.2.4 Other power sources

For equipment using other power sources or capable of being operated from a variety of power sources (primary or secondary), the extreme test voltages shall be those declared by the manufacturer. These shall be recorded in the test report.

7.7 Test condition parameters

The radio parameters shall be tested in the conditions shown in Table 11.

Table 11—Test conditions

Parameter	Temperature	Power source
Output power	ETC[a]	ETC
Power control	NTC[b]	NTC
Modulation index	ETC	ETC
Initial carrier frequency accuracy	ETC	ETC
Carrier frequency drift	ETC	ETC
Conducted in-band spurious emissions	ETC	ETC
Radiated in-band emissions	NTC	NTC
Sensitivity	ETC	ETC
Interference performance	NTC	NTC
Intermodulation characteristics	NTC	NTC
Out-of-band blocking	NTC	NTC
Maximum usable level	NTC	NTC
RSSI	NTC	NTC

[a]ETC = extreme test conditions.
[b]NTC = nominal test conditions.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8. Baseband (BB)

This clause describes the protocols and other lower level link routines that bridge between the PHY and upper level protocols.

8.1 General description

This standard provides a point-to-point connection or a point-to-multipoint connection (see a and b in Figure 8). In a point-to-point connection, the physical channel is shared between two IEEE 802.15.1-2005 devices. In a point-to-multipoint connection, the physical channel is shared among several devices. Two or more devices sharing the same physical channel form a *piconet*. One device acts as the master of the piconet, whereas the other device(s) act as slave(s). Up to seven slaves can be active in the piconet. Additionally, many more slaves can remain connected in PARK state. These parked slaves are not active on the channel, but remain synchronized to the master and can become active without using the connection establishment procedure. Both for active and parked slaves, the channel access is controlled by the master.

Piconets that have common devices are called a *scatternet* (see c in Figure 8). Each piconet has only a single master; however, slaves can participate in different piconets on a TDM basis. In addition, a master in one piconet can be a slave in other piconets. Piconets shall not be frequency synchronized, and each piconet has its own hopping sequence.

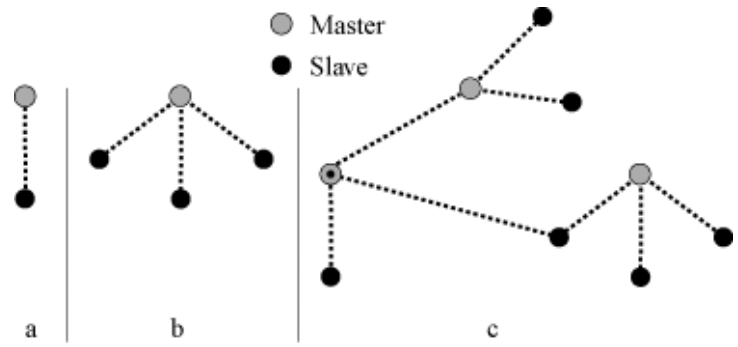


Figure 8—Piconets with a single slave operation (a), a multislave operation (b) and a scatternet operation (c)

Data are transmitted over the air in packets. The general packet format is shown in Figure 9. Each packet consists of 3 entities: the access code, the header, and the payload.

ACCESS CODE	HEADER	PAYLOAD

Figure 9—Standard packet format

8.1.1 Clock

Every IEEE 802.15.1-2005 device shall have a CLKN that shall be derived from a free-running system clock. For synchronization with other devices, offsets are used that, when added to the CLKN, provide temporary clocks that are mutually synchronized. It should be noted that the master clock (CLK) has no relation to the time of day; therefore, it may be initialized to any value. The clock has a cycle of about a day. If the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

clock is implemented with a counter, a 28-bit counter is required that shall wrap around at $2^{28} - 1$. The least significant bit (LSB) shall tick in units of 312.5 μs (i.e., half a time slot), giving a clock rate of 3.2 kHz.

The clock determines critical periods and triggers the events in the device. Four periods are important in the IEEE 802.15.1-2005 system: 312.5 μs, 625 μs, 1.25 ms, and 1.28 s. These periods correspond to the timer bits CLK_0, CLK_1, CLK_2, and CLK_{12}, respectively (see Figure 10).

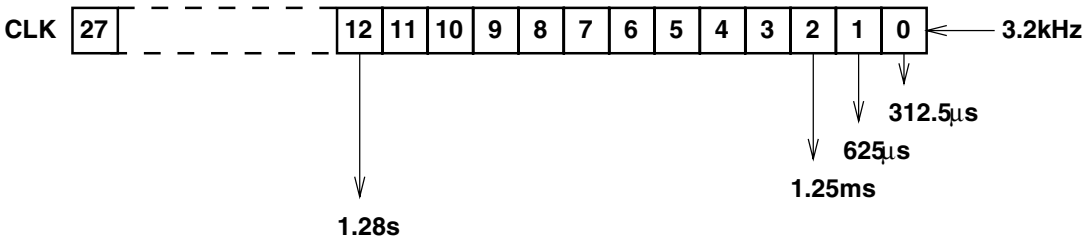


Figure 10—IEEE 802.15.1-2005 clock

In the different modes and states in which a device can reside, the clock has different appearances:

— CLKN native clock
— CLKE estimated clock
— CLK master clock

CLKN is the native clock and shall be the reference to all other clock appearances. In STANDBY and PARK states and in HOLD and SNIFF modes, the CLKN may be driven by a low-power oscillator (LPO) with worst-case accuracy (± 250 ppm). Otherwise, the CLKN shall have a worst-case accuracy of ± 20 ppm.

See 8.2.2.4 for the definition of CLK and 8.2.4.1 for the definition of CLKE.

The master shall never adjust its CLKN during the existence of the piconet.

8.1.2 Device addressing

Each IEEE 802.15.1-2005 device shall be allocated a unique 48-bit device address (BD_ADDR). This address shall be obtained from the IEEE Registration Authority. The address is divided into the following three fields:

— LAP field: lower address part consisting of 24 bits
— UAP field: upper address part consisting of 8 bits
— NAP field: nonsignificant address part consisting of 16 bits

The LAP and UAP form the significant part of the BD_ADDR. The bit pattern in Figure 11 is an example of a BD_ADDR.

The BD_ADDR may take any values except the 64 reserved LAP values for general and dedicated inquiries (see 8.1.2.1).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

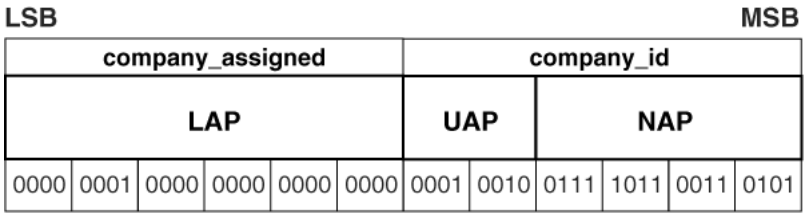


Figure 11—Format of BD_ADDR

8.1.2.1 Reserved addresses

A block of 64 contiguous LAPs is reserved for inquiry operations. One LAP common to all devices is reserved for general inquiry, and the remaining 63 LAPs are reserved for dedicated inquiry of specific classes of devices (see Bluetooth Assigned Numbers [B1]). The same LAP values are used regardless of the contents of UAP and NAP. Consequently, none of these LAPs can be part of a user BD_ADDR.

The reserved LAP addresses are 0x9E8B00–0x9E8B3F. The general inquiry LAP is 0x9E8B33. All addresses have the LSB at the rightmost position, hexadecimal notation. The default check initialization (DCI) is used as the UAP whenever one of the reserved LAP addresses is used. The DCI is defined to be 0x00 (hexadecimal).

8.1.3 Access codes

In IEEE 802.15.1, all transmissions over the physical channel begin with an access code. Three different access codes are defined (see also 8.6.3.1):

— Device access code (DAC)
— Channel access code (CAC)
— Inquiry access code (IAC)

All access codes are derived from the LAP of a device address or an inquiry address. The DAC is used during the **page**, the **page scan**, and the two page response substates and shall be derived from the paged device's BD_ADDR. The CAC is used in the CONNECTION state and forms the beginning of all packets exchanged on the piconet physical channel. The CAC shall be derived from the LAP of the master's BD_ADDR. Finally, the IAC shall be used in the **inquiry** substate. There is one general IAC (GIAC) for general inquiry operations, and there are 63 dedicated IACs (DIACs) for dedicated inquiry operations.

The access code also indicates to the receiver the arrival of a packet. It is used for timing synchronization and offset compensation. The receiver correlates against the entire synchronization word in the access code, providing very robust signalling.

8.2 Physical channels

The lowest architectural layer in the IEEE 802.15.1-2005 system is the physical channel. A number of types of physical channel are defined. All physical channels are characterized by the combination of a pseudo-random frequency hopping sequence, the specific slot timing of the transmissions, the access code, and packet header encoding. These aspects, together with the range of the transmitters, define the signature of the physical channel. For the basic and adapted piconet physical channels, frequency hopping is used to change frequency periodically to reduce the effects of interference and to satisfy local regulatory requirements.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Two devices that wish to communicate use a shared physical channel for this communication. To achieve this, their transceivers must be tuned to the same RF at the same time, and they must be within a nominal range of each other.

Given that the number of RF carriers is limited and that many IEEE 802.15.1-2005 devices may be operating independently within the same spatial and temporal area, there is a strong likelihood of two independent devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision. To mitigate the unwanted effects of this collision, each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This CAC is a property of the physical channel. The access code is always present at the start of every transmitted packet.

Four physical channels are defined. Each is optimized and used for a different purpose. Two of these physical channels (i.e., the basic and adapted piconet physical channels) are used for communication between connected devices and are associated with a specific piconet. The remaining physical channels are used for discovering (i.e., the inquiry scan physical channel) and connecting (i.e., the page scan physical channel) devices.

An IEEE 802.15.1-2005 device can use only one of these physical channels at any given time. In order to support multiple concurrent operations, the device uses TDM between the channels. In this way, a device can appear to operate simultaneously in several piconets as well as being discoverable and connectable.

Whenever an IEEE 802.15.1-2005 device is synchronized to the timing, frequency, and access code of a physical channel, it is said to be *connected* to this channel (whether or not it is actively involved in communications over the channel). At a minimum, a device need be capable of connection to only one physical channel at a time; however, advanced devices may be capable of connecting simultaneously to more than one physical channel, but this standard does not assume that this is possible.

8.2.1 Physical channel definition

Physical channels are defined by a pseudo-random RF channel hopping sequence, the packet (slot) timing, and an access code. The hopping sequence is determined by the UAP and LAP of a device address and the selected hopping sequence. The phase in the hopping sequence is determined by the CLK. All physical channels are subdivided into time slots whose length is different depending on the physical channel. Within the physical channel, each reception or transmission event is associated with a time slot or time slots. For each reception or transmission event, an RF channel is selected by the hop selection kernel (see 8.2.6). The maximum hop rate is 1600 hop/s in the CONNECTION state and the maximum is 3200 hop/s in the **inquiry** and **page** substates.

The following physical channels are defined:

— Basic piconet physical channel
— Adapted piconet physical channel
— Page scan physical channel
— Inquiry scan physical channel

8.2.2 Basic piconet physical channel

During the CONNECTION state, the basic piconet physical channel is used by default. The adapted piconet physical channel may also be used. The adapted piconet physical channel is identical to the basic piconet physical channel except for the differences listed in 8.2.3.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.2.2.1 Master-slave definition

The basic piconet physical channel is defined by the master of the piconet. The master controls the traffic on the piconet physical channel by a polling scheme (see 8.8.5).

By definition, the device that initiates a connection by paging is the master. Once a piconet has been established, master-slave roles may be exchanged. This is described in 8.8.6.5.

8.2.2.2 Hopping characteristics

The basic piconet physical channel is characterized by a pseudo-random hopping through all 79 RF channels. The frequency hopping in the piconet physical channel is determined by CLKN and BD_ADDR of the master. When the piconet is established, the CLK is communicated to the slaves. Each slave shall add an offset to its CLKN to synchronize with the CLK. Since the clocks are independent, the offsets must be updated regularly. All devices participating in the piconet are time-synchronized and hop-synchronized to the channel.

The basic piconet physical channel uses the basic channel hopping sequence, which is described in 8.2.6.

8.2.2.3 Time slots

The basic piconet physical channel is divided into time slots, each 625 μs in length. The time slots are numbered according to the 27 most significant bits (MSBs) of the clock CLK_{28-1} of the piconet master. The slot numbering ranges from 0 to $2^{27} - 1$ and is cyclic with a cycle length of 2^{27}. The time slot number is denoted as *k*.

A TDD scheme is used where master and slave alternatively transmit (see Figure 12). The packet start shall be aligned with the slot start. Packets may extend over up to five time slots.

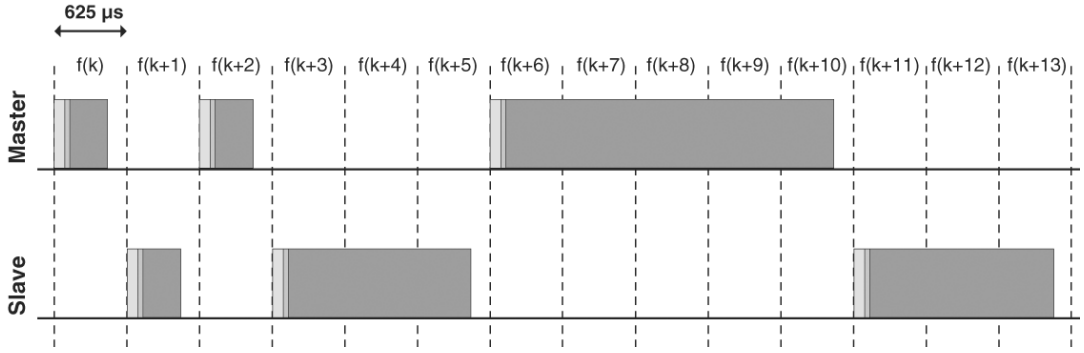


Figure 12—Multislot packets

The term *slot pairs* is used to indicate two adjacent time slots starting with a master-to-slave transmission slot.

8.2.2.4 Piconet clocks

CLK is the master clock of the piconet. It shall be used for all timing and scheduling activities in the piconet. All devices shall use the CLK to schedule their transmission and reception. The CLK shall be derived from the CLKN (see 8.1.1) by adding an offset (see Figure 13). The offset shall be zero for the master since CLK is identical to its own CLKN. Each slave shall add an appropriate offset to its CLKN so that the CLK

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

corresponds to the CLKN of the master. Although all CLKNs in the devices run at the same nominal rate, mutual drift causes inaccuracies in CLK. Therefore, the offsets in the slaves must be regularly updated so that CLK is approximately the CLKN of the master.

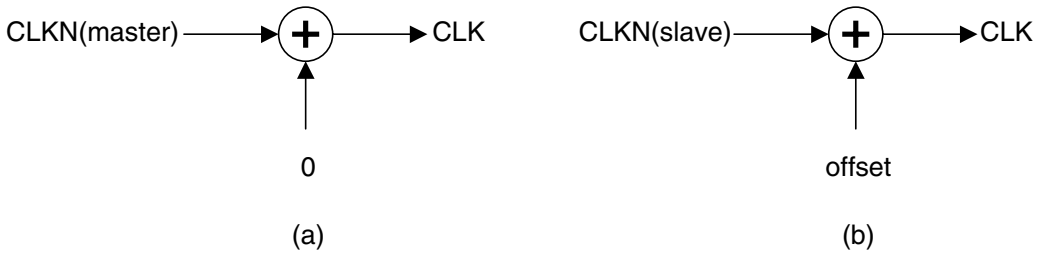


Figure 13—Derivation of CLK in master (a) and in slave (b)

8.2.2.5 Transmit/receive timing

The master transmission shall always start at even-numbered time slots ($CLK_1 = 0$), and the slave transmission shall always start at odd-numbered time slots ($CLK_1 = 1$). Due to packet types that cover more than a single slot, master transmission may continue in odd-numbered slots, and slave transmission may continue in even-numbered slots (see Figure 12).

All timing figures shown in this clause are based on the signals as present at the antenna. The term *exact* when used to describe timing refers to an ideal transmission or reception and neglects timing jitter and clock frequency imperfections.

The average timing of packet transmission shall not drift faster than 20 ppm relative to the ideal slot timing of 625 μs. The instantaneous timing shall not deviate more than 1 μs from the average timing. Thus, the absolute packet transmission timing t_k of slot boundary k shall fulfill Equation (1).

$$t_k = \left(\sum_{i=1}^{k} (1 + d_i) T_N \right) + j_k + \text{offset}, \tag{1}$$

where

T_N is the nominal slot length (625 μs);

j_k denotes jitter ($|j_k| \leq 1$ μs) at the start of slot k;

d_k denotes the drift ($|d_k| \leq 20$ ppm) within slot k.

The jitter and drift may vary arbitrarily within the given limits for every slot, while offset is an arbitrary, but fixed, constant. For HOLD mode, PARK state, and SNIFF mode, the drift and jitter parameters specified in 9.3.3.1 apply.

8.2.2.5.1 Piconet physical channel timing

In the figures, only single-slot packets are shown as an example.

The master TX/RX timing is shown in Figure 14. In Figure 14 and Figure 15, the channel hopping frequencies are indicated by $f(k)$ where k is the time slot number. After transmission, a return packet is expected

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

$N \times 625$ μs after the start of the TX packet where N is an odd integer larger than 0. N depends on the type of the transmitted packet.

To allow for some time slipping, an uncertainty window is defined around the exact receive timing. During normal operation, the window length shall be 20 μs, which allows the RX packet to arrive up to 10 μs too early or 10 μs too late. It is recommended that slaves implement variable-sized windows or time tracking to accommodate a master's absence of more than 250 ms.

During the beginning of the RX cycle, the access correlator shall search for the correct CAC over the uncertainty window. If an event trigger does not occur, the receiver may go to sleep until the next RX event. If, in the course of the search, it becomes apparent that the correlation output will never exceed the final threshold, the receiver may go to sleep earlier. If a trigger event occurs, the receiver shall remain open to receive the rest of the packet unless the packet is for another device, a nonrecoverable header error is detected, or a nonrecoverable payload error is detected.

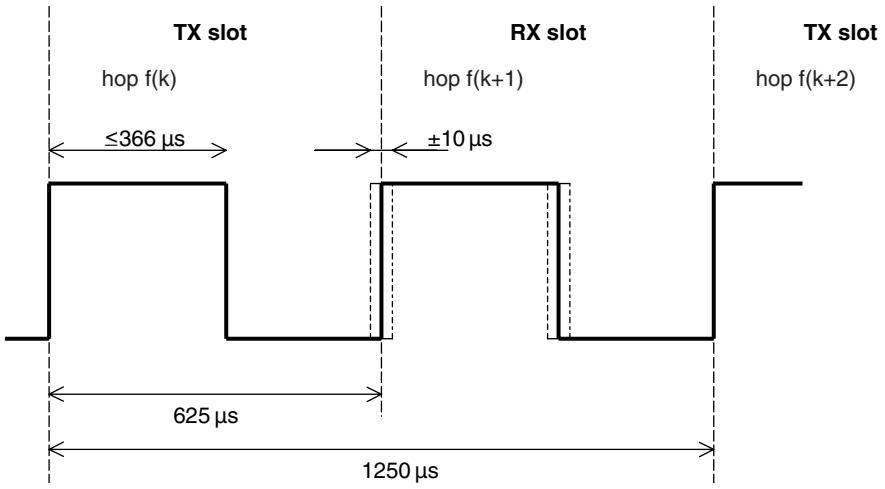


Figure 14—RX/TX cycle of master transceiver in normal mode for single-slot packets

Each master transmission shall be derived from bit 2 of the CLK; thus, the current transmission will be scheduled $M \times 1250$ μs after the start of the previous master TX burst where M depends on the transmitted and received packet type and is an even integer larger than 0. The master TX timing shall be derived from CLK, and thus it will not be affected by time drifts in the slave(s).

Slaves maintain an estimate of CLK by adding a timing offset to the slave's CLKN (see 8.2.2.4). This offset shall be updated each time a packet is received from the master. By comparing the exact RX timing of the received packet with the estimated RX timing, slaves shall correct the offset for any timing misalignments. Since only the CAC is required to synchronize the slave, slave timing can be corrected with any packet sent with the correct CAC.

The slave's TX/RX timing is shown in Figure 15. The slave's transmission shall be scheduled $N \times 625$ μs after the start of the slave's RX packet where N is an odd, positive integer larger than 0. If the slave's RX timing drifts, so will its TX timing. During periods when a slave is in the active mode (see 8.8.6) and is not able to receive any valid CACs from the master, the slave may increase its receive uncertainty window and/or use predicted timing drift to increase the probability of receiving the master's bursts when reception resumes.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

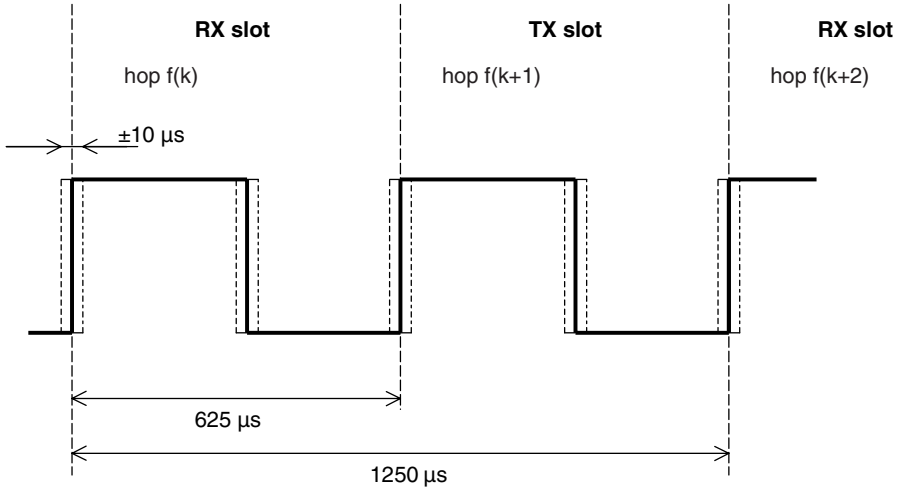


Figure 15—RX/TX cycle of slave transceiver for single-slot packets

8.2.2.5.2 Piconet physical channel resynchronization

In the piconet physical channel, a slave may lose synchronization if it does not receive a packet from the master at least every 250 ms (or less if the low-power clock is used). This may occur in SNIFF mode, HOLD mode , PARK state, or a scatternet or due to interference. When resynchronizing to the piconet physical channel, a slave device shall listen for the master before it may send information. In this case, the length of the search window in the slave device may be increased from 20 µs to a larger value X µs as illustrated in Figure 16. Note that only RX hop frequencies are used. The hop frequency used in the master-to-slave (RX) slot shall also be used in the uncertainty window, even when it is extended into the preceding time interval normally used for the slave-to-master (TX) slot.

The slave is using AFH transmit on the master's transmit frequency and using the same CAC as the master. When a resynchronizing slave has a receive window of more than one-slot duration, it could falsely synchronize on a slave transmission instead of a master transmission. To avoid this, when receive windows are wider than a single slot, the slave may check that the whitening pattern in use corresponds to CLKE (i.e., the slave's estimate of the master's CLKN). As the whitening pattern changes with the master's clock value, this allows a slave to distinguish master transmissions even when a slave transmits on the master's transmit frequency.

If the length of search window X exceeds 1250 µs, consecutive windows shall avoid overlapping search windows. Consecutive windows should instead be centered at $f(k)$, $f(k + 4)$, ... $f(k + 4i)$ (where i is an integer), which gives a maximum value $X = 2500$ µs, or even at $f(k)$, $f(k + 6)$, ..$f(k + 6i)$, which gives a maximum value $X = 3750$ µs. The RX hop frequencies used shall correspond to the master-to-slave transmission slots.

It is recommended that single-slot packets be transmitted by the master during slave resynchronization.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

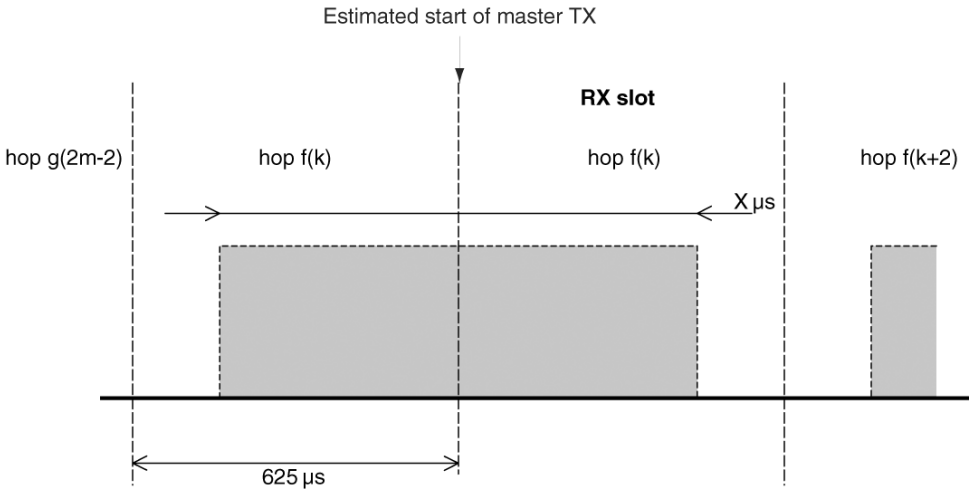


Figure 16—RX timing of slave returning from HOLD mode

8.2.3 Adapted piconet physical channel

8.2.3.1 Hopping characteristics

The adapted piconet physical channel shall use at least N_{min} RF channels (where N_{min} is 20).

The adapted piconet physical channel uses the adapted channel hopping sequence described in 8.2.6.

Adapted piconet physical channels can be used for connected devices that have AFH enabled. There are two differences between basic and adapted piconet physical channels:

— The slave uses the same frequency as the preceding master transmission when AFH is in effect.
— AFH uses less than the full 79 frequencies that the basic piconet uses.

8.2.4 Page scan physical channel

Although master and slave roles are not defined prior to a connection, the term *master* is used for the paging device (that becomes a master in the CONNECTION state), and *slave* is used for the page scanning device (that becomes a slave in the CONNECTION state).

8.2.4.1 Clock estimate for paging

A paging device uses an estimate of the CLKN of the page scanning device, CLKE, i.e., an offset shall be added to the CLKN of the pager to approximate the CLKN of the recipient (see Figure 17). CLKE shall be derived from the reference CLKN by adding an offset. By using the CLKN of the recipient, the pager might be able to speed up the connection establishment.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

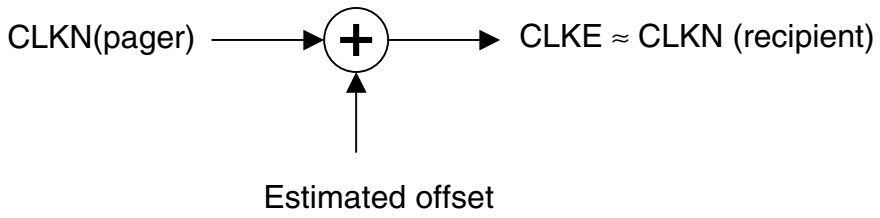


Figure 17—Derivation of CLKE

8.2.4.2 Hopping characteristics

The page scan physical channel follows a slower hopping pattern than the basic piconet physical channel and is a short pseudo-random hopping sequence through the RF channels. The timing of the page scan physical channel shall be determined by CLKN of the scanning device. The frequency hopping sequence is determined by the address of the scanning device.

The page scan physical channel uses the page, master page response, slave page response, and page scan hopping sequences specified in 8.2.6.

8.2.4.3 Paging procedure timing

During the paging procedure, the master shall transmit paging messages (see Table 26) corresponding to the slave to be connected. Since the paging message is a very short packet, the hop rate is 3200 hop/s. In a single TX slot interval, the paging device shall transmit on two different hop frequencies. In Figure 18 through Figure 22, $f(k)$ is used for the frequencies of the page hopping sequence and $f'(k)$ denotes the corresponding page response sequence frequencies. The first transmission starts where $CLK_0 = 0$, and the second transmission starts where $CLK_0 = 1$.

In a single RX slot interval, the paging device shall listen for the slave page response message on two different hop frequencies. Similar to transmission, the nominal reception starts where $CLK_0 = 0$, and the second reception nominally starts where $CLK_0 = 1$ (see Figure 18). During the TX slot, the paging device shall send the paging message at the TX hop frequencies $f(k)$ and $f(k + 1)$. In the RX slot, it shall listen for a response on the corresponding RX hop frequencies $f'(k)$ and $f'(k + 1)$. The listening periods shall be exactly timed 625 μs after the corresponding paging packets and shall include a ± 10 μs uncertainty window.

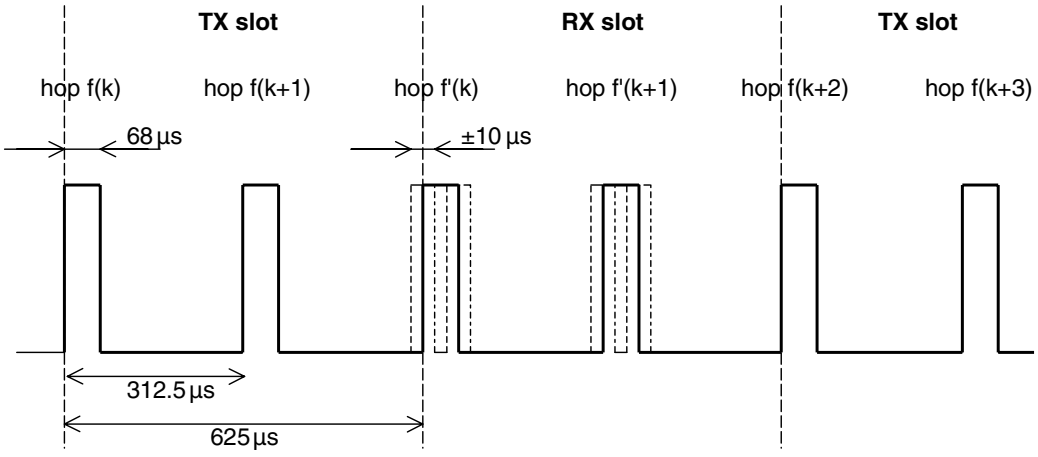


Figure 18—RX/TX cycle of transceiver in page mode

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.2.4.4 Page response timing

At connection setup, a master page response packet is transmitted from the master to the slave (see Table 26). This packet establishes the timing and frequency synchronization. After the slave device has received the page message, it shall return a response message that consists of the slave page response packet and shall follow 625 μs after the receipt of the page message. The master shall send the master page response packet in the TX slot following the RX slot in which it received the slave response, according to the RX/TX timing of the master. The time difference between the slave page response and master page response message will depend on the timing of the page message the slave received. In Figure 19, the slave receives the paging message sent **first** in the master-to-slave slot. It then responds with a first slave page response packet in the first half of the slave-to-master slot. The timing of the master page response packet is based on the timing of the page message sent first in the preceding master-to-slave slot: there is an exact 1250 μs delay between the first page message and the master page response packet. The packet is sent at the hop frequency $f(k+1)$, which is the hop frequency following the hop frequency $f(k)$ in which the page message was received.

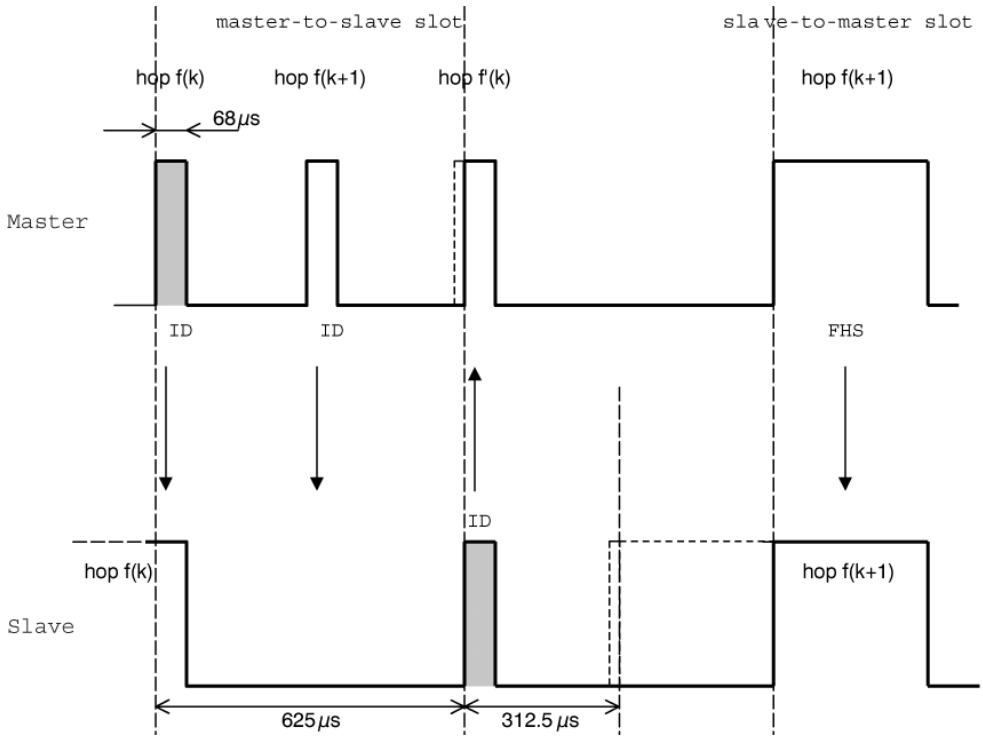


Figure 19—Timing of page response packets on successful page in first half slot

In Figure 20, the slave receives the paging message sent second in the master-to-slave slot. It then responds with a slave page response packet in the second half of the slave-to-master slot exactly 625 μs after the receipt of the page message. The timing of the master page response packet is still based on the timing of the page message sent **first** in the preceding master-to-slave slot: there is an exact 1250 μs delay between the **first** page message and the master page response packet. The packet is sent at the hop frequency $f(k+2)$, which is the hop frequency following the hop frequency $f(k+1)$ in which the page message was received.

The slave shall adjust its RX/TX timing according to the reception of the master page response packet (and not according to the reception of the page message). In other words, the second slave page response message that acknowledges the reception of the master page response packet shall be transmitted 625 μs after the start of the master page response packet.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

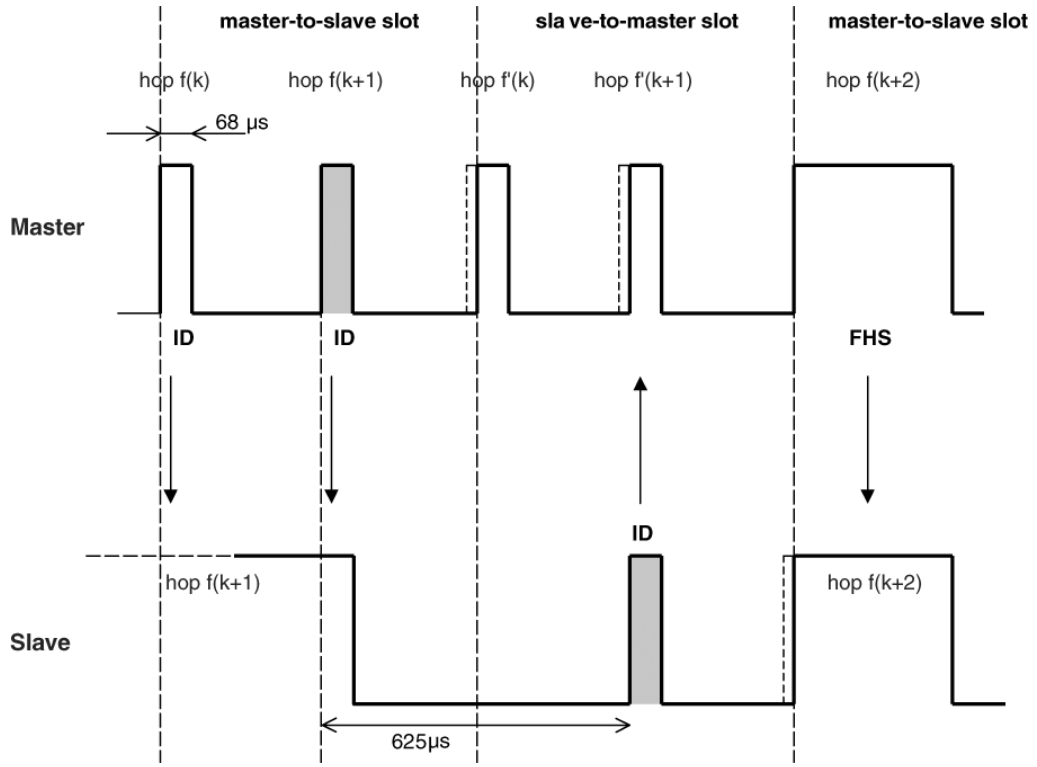


Figure 20—Timing of page response packets on successful page in second half slot

8.2.5 Inquiry scan physical channel

Although master and slave roles are not defined prior to a connection, the term *master* is used for the inquiring device, and *slave* is used for the inquiry scanning device.

8.2.5.1 Clock for inquiry

The clock used for inquiry and inquiry scan shall be the device's CLKN.

8.2.5.2 Hopping characteristics

The inquiry scan physical channel follows a slower hopping pattern than the piconet physical channel and is a short pseudo-random hopping sequence through the RF channels. The timing of the inquiry scan physical channel is determined by the CLKN of the scanning device while the frequency hopping sequence is determined by the GIAC.

The inquiry scan physical channel uses the inquiry, inquiry response, and inquiry scan hopping sequences described in 8.2.6.

8.2.5.3 Inquiry procedure timing

During the inquiry procedure, the master shall transmit inquiry messages with the GIAC or DIAC. The timing for inquiry is the same as for paging (see 8.2.4.3).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.2.5.4 Inquiry response timing

An inquiry response packet is transmitted from the slave to the master after the slave has received an inquiry message (see Table 28). This packet contains information necessary for the inquiring master to page the slave (see definition of the FHS packet in 8.6.5.1.4) and follows 625 µs after the receipt of the inquiry message. In Figure 21 and Figure 22, $f(k)$ is used for the frequencies of the inquiry hopping sequence and $f'(k)$ denotes the corresponding inquiry response sequence frequency. The packet is received by the master at the hop frequency $f'(k)$ when the inquiry message received by the slave was first in the master-to-slave slot.

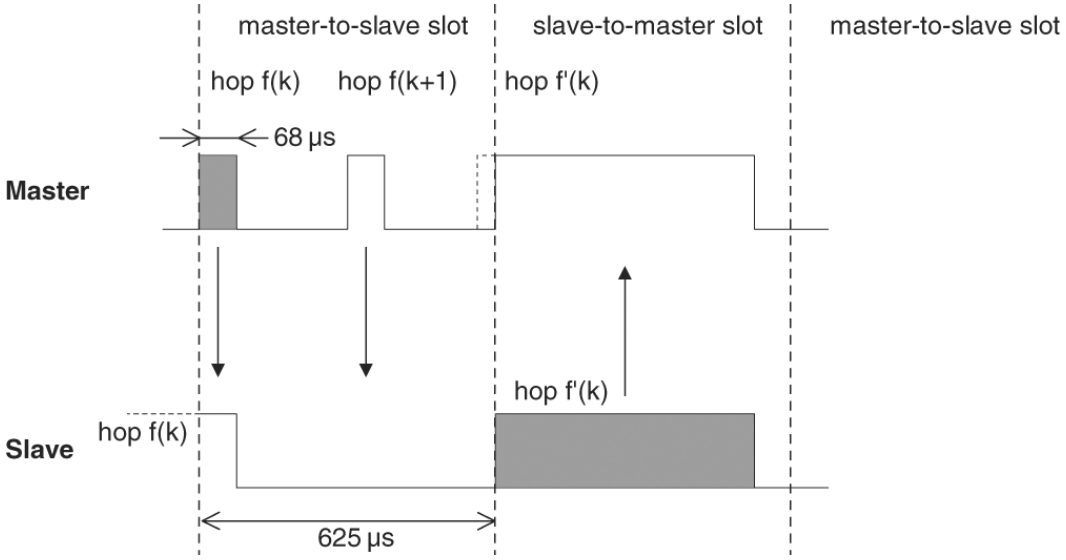


Figure 21—Timing of inquiry response packet on successful inquiry in first half slot

When the inquiry message received by the slave was the second in the master-to-slave slot, the packet is received by the master at the hop frequency $f'(k + 1)$.

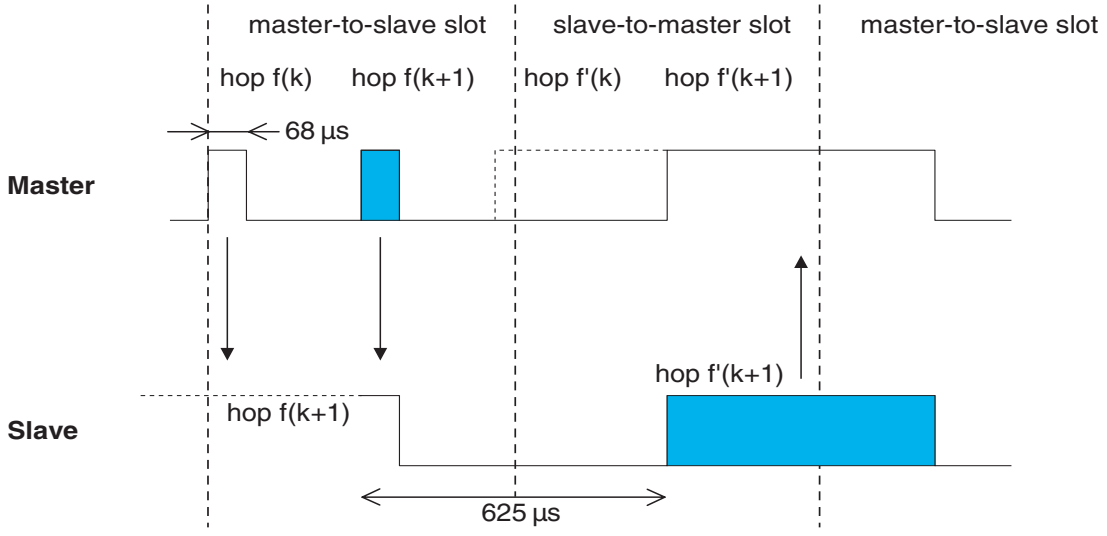


Figure 22—Timing of inquiry response packet on successful inquiry in second half slot

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.2.6 Hop selection

IEEE 802.15.1-2005 devices shall use the hopping kernel as defined in 8.2.6.2 and 8.2.6.3.

In total, six types of hopping sequence are defined: five for the basic hop system and one for an adapted set of hop locations used by AFH. These sequences are as follows:

— A *page hopping sequence* with 32 wake-up frequencies distributed equally over the 79 MHz, with a period length of 32.
— A *page response hopping sequence* covering 32 response frequencies that are in a one-to-one correspondence to the current page hopping sequence. The master and slave use different rules to obtain the same sequence.
— An *inquiry hopping sequence* with 32 wake-up frequencies distributed equally over the 79 MHz, with a period length of 32.
— An *inquiry response hopping sequence* covering 32 response frequencies that are in a one-to-one correspondence to the current inquiry hopping sequence.
— A *basic channel hopping sequence*, which has a very long period length, which does not show repetitive patterns over a short time interval, and which distributes the hop frequencies equally over the 79 MHz during a short time interval.
— An *adapted channel hopping sequence*, derived from the basic channel hopping sequence, which uses the same channel mechanism and may use fewer than 79 frequencies. The adapted channel hopping sequence is used only in place of the basic channel hopping sequence. All other hopping sequences are not affected by hop sequence adaptation.

8.2.6.1 General selection scheme

The selection scheme consists of two parts:

— Selecting a sequence
— Mapping this sequence onto the hop frequencies

The general block diagram of the hop selection scheme is shown in Figure 23. The mapping from the input to a particular RF channel index is performed in the selection box.

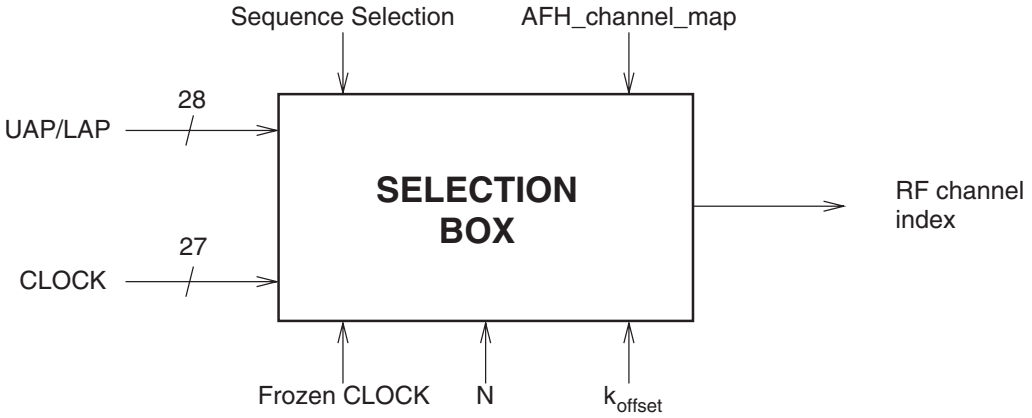


Figure 23—General block diagram of hop selection scheme

The inputs to the selection box are the selected clock, frozen clock, N, k_{offset}, address, sequence selection and AFH_channel_map. The source of the clock input depends on the hopping sequence selected. Additionally, each hopping sequence uses different bits of the clock (see Table 13). N and k_{offset} are defined in 8.2.6.4.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The sequence selection input can be set to the following values:

— Page scan
— Inquiry scan
— Page
— Inquiry
— Master page response
— Slave page response
— Inquiry response
— Basic channel
— Adapted channel

The address input consists of 28 bits including the entire LAP and the 4 LSBs of the UAP. This is designated as the UAP/LAP. When the basic or adapted channel hopping sequence is selected, the device address of the master (BD_ADDR) shall be used. When the page, master page response, slave page response, or page scan hopping sequences are selected, the BD_ADDR given by the host of the paged device shall be used (see 11.7.1.5). When the inquiry, inquiry response, or inquiry scan hopping sequences are selected, the UAP/LAP corresponding to the GIAC shall be used even if it concerns a DIAC. Whenever one of the reserved BD_ADDRs (see 8.1.2.1) is used for generating a frequency hop sequence, the UAP shall be replaced by the DCI (see 8.7.1). The hopping sequence is selected by the sequence selection input to the selection box.

When the adapted channel hopping sequence is selected, the AFH_channel_map is an additional input to the selection box. The AFH_channel_map indicates which channels shall be used and which shall be unused. These terms are defined in 8.2.6.3.

The output *RF channel index* constitutes a pseudo-random sequence. The RF channel index is mapped to RF channel frequencies using the equation in Table 4 (in Clause 7).

The selection scheme chooses a segment of 32 hop frequencies spanning about 64 MHz and visits these hops in a pseudo-random order. Next, a different 32-hop segment is chosen, etc. In the page, master page response, slave page response, page scan, inquiry, inquiry response, and inquiry scan hopping sequences, the same 32-hop segment is used all the time. (The segment is selected by the address; different devices will have different paging segments.) When the basic channel hopping sequence is selected, the output constitutes a pseudo-random sequence that slides through the 79 hops. The principle is depicted in Figure 24.

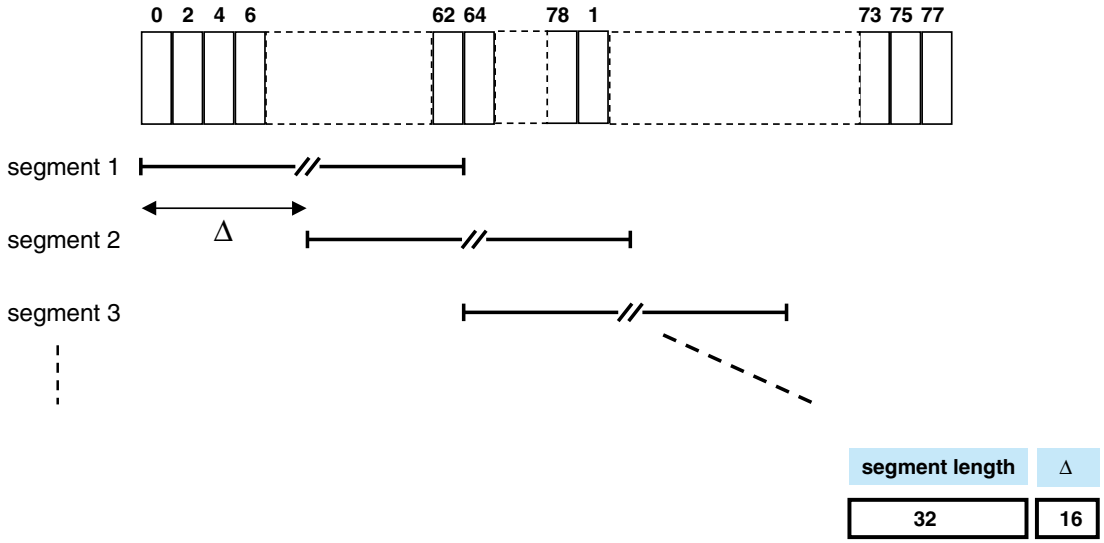


Figure 24—Hop selection scheme in CONNECTION state

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The RF shall remain fixed for the duration of the packet. The RF for the packet shall be derived from the CLK value in the first slot of the packet. The RF in the first slot after a multislot packet shall use the frequency as determined by the CLK value for that slot. Figure 25 illustrates the hop definition on single-slot and multislot packets.

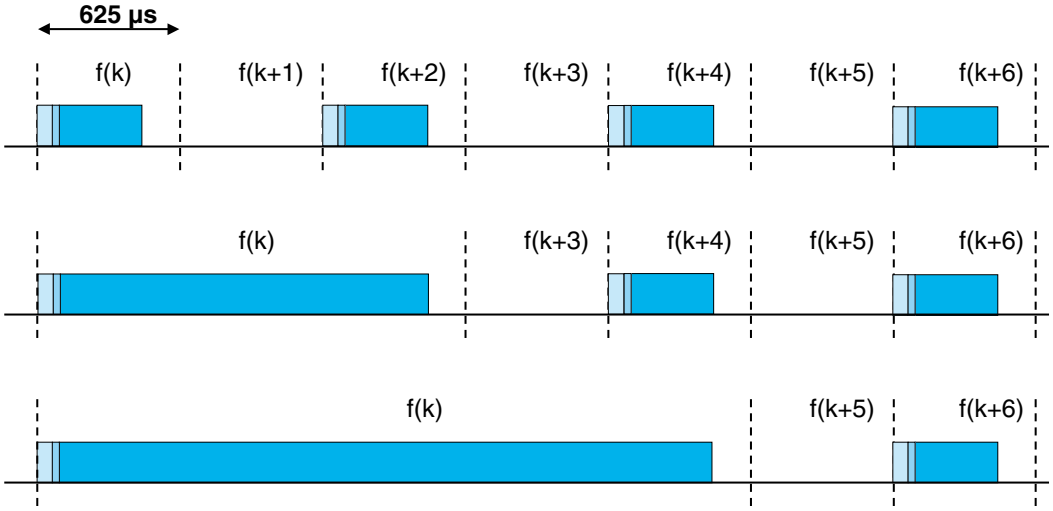


Figure 25—Single- and multislot packets

When the adapted channel hopping sequence is used, the pseudo-random sequence contains only frequencies that are in the RF channel set defined by the input AFH_channel_map. The adapted sequence has similar statistical properties to the nonadapted hop sequence (non-AHS). In addition, the slave responds with its packet on the same RF channel that was used by the master to address that slave (or would have been in the case of a synchronous reserved slot without a validly received master-to-slave transmission). This is called the *same channel mechanism* of AFH. Thus, the RF channel used for the master-to-slave packet is also used for the immediately following slave-to-master packet. An example of the same channel mechanism is illustrated in Figure 26. The same channel mechanism shall be used whenever the adapted channel hopping sequence is selected.

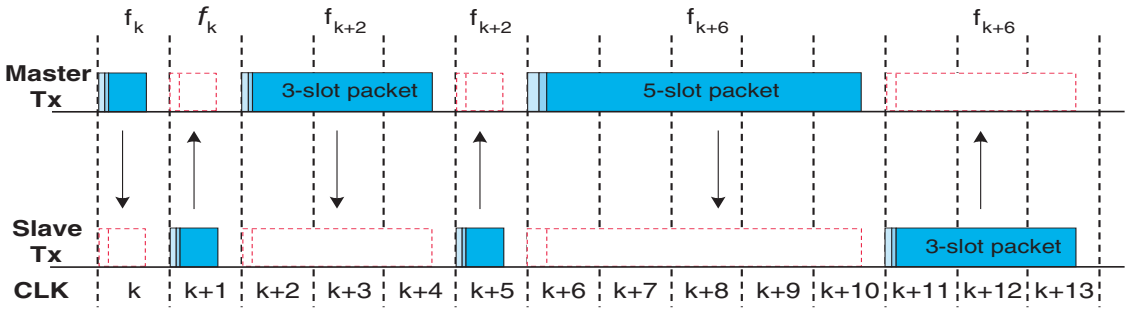


Figure 26—Example of the same channel mechanism

8.2.6.2 Basic hop selection kernel

The basic hop selection kernel shall be as shown in Figure 27 and is used for the page, page response, inquiry, inquiry response, and basic channel hopping selection kernels. In these substates, the input AFH_channel_map is unused. The adapted hop selection kernel is described in 8.2.6.3. The X input determines the phase in the 32-hop segment, whereas Y1 and Y2 select between master to slave and slave to master. Inputs A to D determine the ordering within the segment, inputs E and F determine the mapping onto

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

the hop frequencies. The kernel addresses a register containing the RF channel indices. This list is ordered so that first all even RF channel indices are listed and then all odd hop frequencies. In this way, a 32-hop segment spans about 64 MHz.

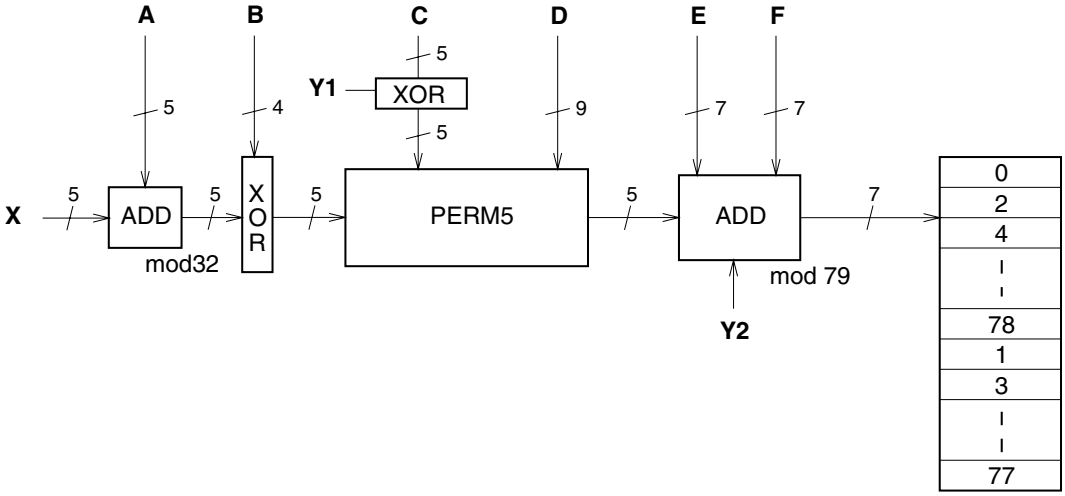


Figure 27—Block diagram of the basic hop selection kernel for the hop system

The selection procedure consists of an addition, an XOR operation, a permutation operation, an addition, and finally a register selection. In the remainder of this clause, the notation A_i is used for bit i of the BD_ADDR.

8.2.6.2.1 First addition operation

The first addition operation adds a constant only to the phase and applies a modulo-32 operation. For the page hopping sequence, the first addition is redundant since it changes only the phase within the segment. However, when different segments are concatenated (as in the basic channel hopping sequence), the first addition operation will have an impact on the resulting sequence.

8.2.6.2.2 XOR operation

Let Z' denote the output of the first addition. In the XOR operation, the 4 LSBs of Z' are modulo-2 added to the address bits $A_{22–19}$. The operation is illustrated in Figure 28.

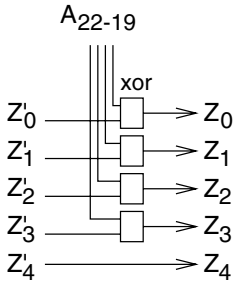


Figure 28—XOR operation for hop system

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.2.6.2.3 Permutation operation

The permutation operation involves the switching from five inputs to five outputs for the hop system, controlled by the control word. The permutation or switching box shall be as shown in Figure 29. It consists of seven stages of butterfly operations. The control of the butterflies by the control signals P is shown in Table 12. P_{0-8} corresponds to D_{0-8}, and, P_{i+9} corresponds to $C_i \oplus Y1$ for $i = 0...4$ in Figure 27.

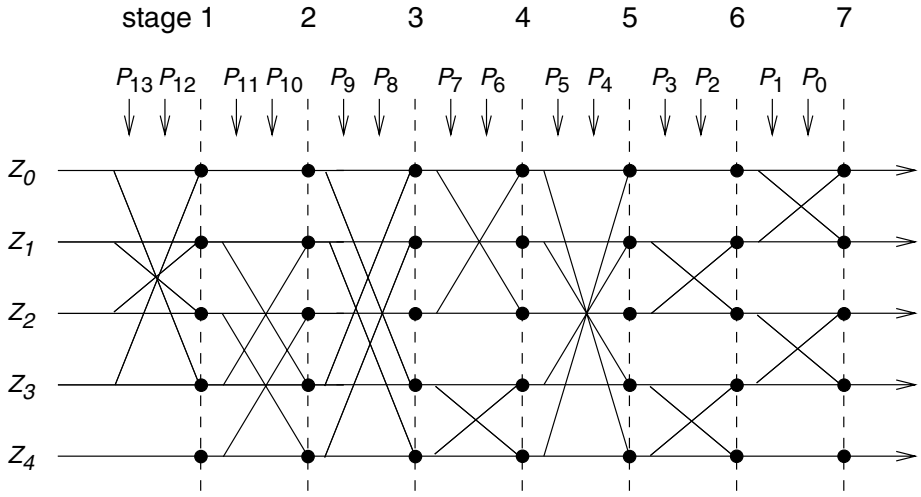


Figure 29—Permutation operation for hop system

Table 12—Control of butterflies for hop system

Control signal	Butterfly		Control signal	Butterfly
P_0	$\{Z_0,Z_1\}$		P_8	{Z1,Z4}
P_1	$\{Z_2,Z_3\}$		P_9	$\{Z_0,Z_3\}$
P_2	$\{Z_1,Z_2\}$		P_{10}	$\{Z_2,Z_4\}$
P_3	$\{Z_3,Z_4\}$		P_{11}	$\{Z_1,Z_3\}$
P_4	$\{Z_0,Z_4\}$		P_{12}	$\{Z_0,Z_3\}$
P_5	$\{Z_1,Z_3\}$		P_{13}	$\{Z_1,Z_2\}$
P_6	$\{Z_0,Z_2\}$			
P_7	$\{Z_3,Z_4\}$			

The Z input is the output of the XOR operation as described in 8.2.6.2.2. The butterfly operation can be implemented with multiplexers as depicted in Figure 30.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

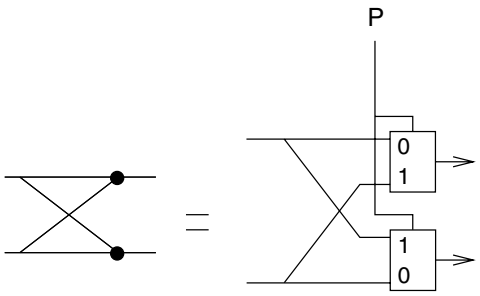


Figure 30—Butterfly implementation

8.2.6.2.4 Second addition operation

The addition operation adds a constant only to the output of the permutation operation. The addition is applied modulo-79.

8.2.6.2.5 Register bank

The output of the adder addresses a bank of 79 registers. The registers are loaded with the synthesizer code words corresponding to the hop frequencies 0 to 78. Note that the upper half of the bank contains the even hop frequencies, whereas the lower half of the bank contains the odd hop frequencies.

8.2.6.3 Adapted hop selection kernel

The adapted hop selection kernel is based on the basic hop selection kernel defined in 8.2.6.2.

The inputs to the adapted hop selection kernel are the same as for the basic hop selection kernel except that the input AFH_channel_map (defined in 9.4.2) is used. The AFH_channel_map indicates which RF channels shall be used and which shall be unused. When hop sequence adaptation is enabled, the number of used RF channels may be reduced from 79 to some smaller value N. All devices shall be capable of operating on an adapted hop sequence (AHS) with $N_{min} \leq N \leq 79$, with any combination of used RF channels within the AFH_channel_map that meets this constraint. N_{min} is defined in 8.2.3.1.

Adaptation of the hopping sequence is achieved through two additions to the basic channel hopping sequence according to Figure 27:

— Unused RF channels are remapped uniformly onto used RF channels. In other words, if the hop selection kernel of the basic system generates an unused RF channel, an alternative RF channel out of the set of used RF channels is selected pseudo-randomly.
— The used RF channel generated for the master-to-slave packet is also used for the immediately following slave-to-master packet (see 8.2.6.1).

8.2.6.3.1 Channel remapping function

When the adapted hop selection kernel is selected, the basic hop selection kernel according to Figure 27 is initially used to determine an RF channel. If this RF channel is unused according to the AFH_channel_map, the unused RF channel is remapped by the remapping function to one of the used RF channels. If the RF channel determined by the basic hop selection kernel is already in the set of used RF channels, no adjustment is made. The hop sequence of the (nonadapted) basic hop equals the sequence of the adapted selection kernel on all locations where used RF channels are generated by the basic hop. This property facilitates non-AFH slaves remaining synchronized while other slaves in the piconet are using the adapted hopping sequence, although it is possible for a resynchronizing slave to mistake the transmissions of another slave that is using AFH for the transmissions of the master (see 8.2.2.5.2).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A block diagram of the remapping mechanism is shown in Figure 31. The remapping function is a post-processing step to the selection kernel from Figure 27, denoted as "Hop selection of basic hop system." The output f_k of the basic hop selection kernel is an RF channel number that ranges between 0 and 78. This RF channel will be either in the set of used RF channels or in the set of unused RF channels.

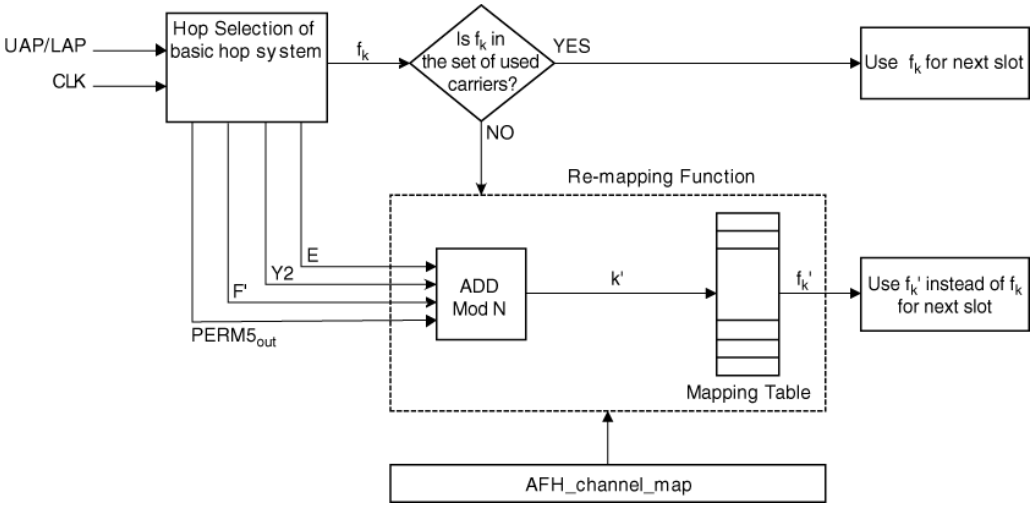


Figure 31—Block diagram of adaptive hop selection mechanism

When an unused RF channel is generated by the basic hop selection mechanism, it is remapped to the set of used RF channels as follows. A new index $k' \in \{0, 1, ..., N-1\}$ is calculated using some of the parameters from the basic hop selection kernel:

$$k' = (PERM5_{out} + E + F' + Y2) \bmod N \quad (2)$$

where F' is defined in Table 13. The index k' is then used to select the remapped channel from a mapping table that contains all of the even used RF channels in ascending order followed by all the odd used RF channels in ascending order (i.e., the mapping table of Figure 27 with all the unused RF channels removed).

8.2.6.4 Control word

In 8.2.6.4 through 8.2.6.4.6, X_{j-i}, $i < j$, will denote bits i, $i + 1$,...,j of the bit vector X. By convention, X_0 is the LSB of the vector X.

The control word of the kernel is controlled by the overall control signals X, Y1, Y2, A to F, and F' as illustrated in Figure 27 and Figure 31. During paging and inquiry, the inputs A to E use the address values as given in the corresponding columns of Table 13. In addition, the inputs X, Y1, and Y2 are used. The inputs F and F' are unused. The clock bits CLK_{6-2} (i.e., input X) specifies the phase within the length 32 sequence. CLK_1 (i.e., inputs Y1 and Y2) is used to select between TX and RX. The address inputs determine the sequence order within segments. The final mapping onto the hop frequencies is determined by the register contents.

During the CONNECTION state (see 8.8.5), the inputs A, C, and D shall be derived from the address bits being bitwise XORed with the clock bits as shown in the "CONNECTION state" column of Table 13. (The 2 MSBs are XORed together; the two second MSBs are XORed together; etc.)

The five X input bits vary depending on the current state of the device. In the **page scan** and **inquiry scan** substates, the CLKN shall be used. In CONNECTION state, the CLK shall be used as input. The situation is somewhat more complicated for the other states.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 13—Control for hop system

	Page scan / interlaced page scan / inquiry scan / interlaced inquiry scan	Page/Inquiry	Master/Slave page response and inquiry response	CONNECTION state
X	CLKN_{16-12} / $(\mathrm{CLKN}_{16-12}+16) mod 32$/ Xir_{4-0} / $Xir_{4-0}+16) mod 32$	Xp_{4-0}/Xi_{4-0}	$Xprm_{4-0}$/ $Xprs_{4-0}$/ Xir_{4-0}	CLK_{6-2}
Y1	0	$\mathrm{CLKE}_1/\mathrm{CLKN}_1$	$\mathrm{CLKE}_1/\mathrm{CLKN}_1/1$	CLK_1
Y2	0	$32\times\mathrm{CLKE}_1$/ $32\times\mathrm{CLKN}_1$	$32\times\mathrm{CLKE}_1$ / $32\times\mathrm{CLKN}_1$ / 32×1	$32\times\mathrm{CLK}_1$
A	A_{27-23}	A_{27-23}	A_{27-23}	$A_{27-23}\oplus\mathrm{CLK}_{25-21}$
B	A_{22-19}	A_{22-19}	A_{22-19}	A_{22-19}
C	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}\oplus\mathrm{CLK}_{20-16}$
D	A_{18-10}	A_{18-10}	A_{18-10}	$A_{18-10}\oplus\mathrm{CLK}_{15-7}$
E	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$
F	0	0	0	$16\times\mathrm{CLK}_{27-7}$ mod 79
F'	n/a	n/a	n/a	$16\times\mathrm{CLK}_{27-7}$ mod N

8.2.6.4.1 Page scan and inquiry scan hopping sequences

When the sequence selection input is set to page scan, the device address of the scanning device shall be used as address input. When the sequence selection input is set to inquiry scan, the GIAC LAP and the four LSBs of the DCI (as A_{27-24}) shall be used as address input for the hopping sequence. For the transmitted access code and in the receiver correlator, the appropriate GIAC or DIAC shall be used. The application decides which IAC to use depending on the purpose of the inquiry.

8.2.6.4.2 Page hopping sequence

When the sequence selection input is set to page, the paging device shall start using the **A**-train, i.e., $\{f(k-8), \ldots, f(k), \ldots, f(k+7)\}$, where $f(k)$ is the source's estimate of the current receiver frequency in the paged device. The index k is a function of all the inputs in Figure 27. There are 32 possible paging frequencies within each 1.28 s interval. Half of these frequencies belong to the **A**-train, the rest (i.e., $\{f(k+8), \ldots, f(k+15), f(k-16), \ldots, f(k-9)\}$) belong to the **B**-train. In order to achieve the –8 offset of the **A**-train, a constant of 24 shall be added to the clock bits (which is equivalent to –8 due to the modulo-32 operation). The **B**-train is obtained by setting the offset to 8. A cyclic shift of the order within the trains is also necessary in order to avoid a possible repetitive mismatch between the paging and scanning devices. Thus,

$$Xp = [\mathrm{CLKE}_{16-12} + k_{offset} + (\mathrm{CLKE}_{4-2,0} - \mathrm{CLKE}_{16-12}) \mod 16] \mod 32, \quad (3)$$

where

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

$$k_{offset} = \begin{cases} 24 & \text{A-train,} \\ 8 & \text{B-train.} \end{cases} \quad (4)$$

Alternatively, each switch between the **A**- and **B**-trains may be accomplished by adding 16 to the current value of k_{offset} (originally initialized with 24).

8.2.6.4.3 Slave page response hopping sequence

When the sequence selection input is set to slave page response, in order to eliminate the possibility of losing the link due to discrepancies of the CLKN and the estimate of the master's clock CLKE, the 4 bits CLKN_{16-12} shall be frozen at their current value. The value shall be frozen at the content it has in the slot where the recipient's access code is detected. The CLKN shall not be stopped; it is merely the values of the bits used for creating the input X that are kept fixed for a while. A frozen value is denoted by an asterisk (*) in the discussion in this subclause.

For each response slot, the paged device shall use an input X value one larger (modulo-32) than in the preceding response slot. However, the first response shall be made with the input X kept at the same value as it was when the access code was recognized. Let N be a counter starting at zero. Then, the input X in the $(N+1)^{\text{th}}$ response slot (the first response slot being the one immediately following the page slot now being responded to) of the **slave response** substate is as follows:

$$Xprs = [\mathrm{CLKN}^*_{16-12} + N] \bmod 32 \quad (5)$$

The counter N shall be set to zero in the slot where the slave acknowledges the page (see Figure 62 and Figure 63). Then, the value of N shall be increased by one each time CLKN_1 is set to zero, which corresponds to the start of a master TX slot. The input X shall be constructed this way until the first FHS packet is received and the immediately following response packet has been transmitted. After this, the slave shall enter the CONNECTION state using the parameters received in the FHS packet.

8.2.6.4.4 Master page response hopping sequence

When the sequence selection input is set to master page response, the master shall freeze its slave CLKE to the value that triggered a response from the paged device. It is equivalent to using the values of the clock estimate when receiving the slave response (since only CLKE_1 will differ from the corresponding page transmission). Thus, the values are frozen when the slave identity (ID) packet is received. In addition to the clock bits used, the current value of k_{offset} shall also be frozen. The master shall adjust its input X in the same way the paged device does, i.e., by incrementing this value by one for each time CLKE_1 is set to zero. The first increment shall be done before sending the FHS packet to the paged device. Let N be a counter starting at one. The rule for forming the input X is as follows:

$$\begin{aligned} Xprm = [&\mathrm{CLKE}^*_{16-12} + k_{offset}{}^* + \\ &(\mathrm{CLKE}^*_{4-2,0} - \mathrm{CLKE}^*_{16-12}) \bmod 16 + N] \bmod 32 \end{aligned} \quad (6)$$

The value of N shall be increased each time CLKE_1 is set to zero, which corresponds to the start of a master TX slot.

8.2.6.4.5 Inquiry hopping sequence

When the sequence selection input is set to inquiry, the input X is similar to that used in the page hopping sequence. Since no particular device is addressed, the CLKN of the inquirer shall be used. Moreover, which of the two train offsets is used to start is of no real concern in this state. Consequently,

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

$$Xi = [\mathrm{CLKN}_{16-12} + k_{offset} + (\mathrm{CLKN}_{4-2,0} - \mathrm{CLKN}_{16-12}) \mod 16] \mod 32, \quad (7)$$

where k_{offset} is defined by Equation (4). The initial choice of the offset is arbitrary.

The GIAC LAP and the 4 LSBs of the DCI (as A_{27-24}) shall be used as address input for the hopping sequence generator.

8.2.6.4.6 Inquiry response hopping sequence

The inquiry response hopping sequence is similar to the slave page response hopping sequence with respect to the input X. The clock input shall not be frozen, thus the following equation applies:

$$Xir = [\mathrm{CLKN}_{16-12} + N] \mod 32 \quad (8)$$

Furthermore, the counter N is increased not on CLKN_1 basis, but rather after each FHS packet has been transmitted in response to the inquiry. There is no restriction on the initial value of N as it is independent of the corresponding value in the inquiring unit.

The GIAC LAP and the 4 LSBs of the DCI (as A_{27-24}) shall be used as address input for the hopping sequence generator. The other input bits to the generator shall be the same as for page response.

8.2.6.4.7 Basic and adapted channel hopping sequence

In the basic and adapted channel hopping sequences, the clock bits to use in the basic or adapted hopping sequence generation shall always be derived from the CLK. The address bits shall be derived from the device address of the master.

8.3 Physical links

A physical link represents a BB connection between devices. A physical link is always associated with exactly one physical channel. Physical links have common properties that apply to all logical transports on the physical link.

The properties of the active physical link are as follows:

— Power control (see 9.3.1.3)
— Link supervision (see 8.3.1 and 9.3.1.6)
— Encryption (see 13.4 and 9.3.2.5)
— Channel quality-driven data rate (CQDDR) change (see 9.3.1.7)
— Multislot packet control (see 9.3.1.10)

All of these properties, except power control and CQDDR, may be applied to the parked physical link.

8.3.1 Link supervision

A connection can break down due to various reasons, such as when a device moves out of range or encounters severe interference or a power failure condition. Since this may happen without any prior warning, it is important to monitor the link on both the master and the slave side to avoid possible collisions when the LT_ADDR (see 8.4.2) or parked member address (PM_ADDR) (see 8.4.7.1) is reassigned to another slave.

To be able to detect link loss, both the master and the slave shall use a link supervision timer, $T_{supervision}$. Upon reception of a valid packet header with one of the slave's addresses (see 8.4.2) on the physical link, the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

timer shall be reset. If, at any time in CONNECTION state, the timer reaches the *supervisionTO* value, the connection shall be considered disconnected. The same link supervision timer shall be used for SCO, eSCO, and ACL logical transports.

The timeout period *supervisionTO* is negotiated by the LM. Its value shall be chosen so that the supervision timeout will be longer than HOLD and SNIFF mode periods. Link supervision of a parked slave shall be done by unparking and reparking the slave.

8.4 Logical transports

8.4.1 General

Between master and slave(s), different types of logical transports may be established. Five logical transports have been defined:

— Synchronous connection-oriented (SCO) logical transport
— Extended synchronous connection-oriented (eSCO) logical transport
— Asynchronous connection-oriented (ACL) logical transport
— Active slave broadcast (ASB) logical transport
— Parked slave broadcast (PSB) logical transport

The synchronous logical transports are point-to-point logical transports between a master and a single slave in the piconet. The synchronous logical transports typically support time-bounded information like voice or general synchronous data. The master maintains the synchronous logical transports by using reserved slots at regular intervals. In addition to the reserved slots, the eSCO logical transport may have a retransmission window after the reserved slots.

The ACL logical transport is also a point-to-point logical transport between the master and a slave. In the slots not reserved for synchronous logical transport(s), the master can establish an ACL logical transport on a per-slot basis to any slave, including the slave(s) already engaged in a synchronous logical transport. There are cases when the ACL logical transport may use slots reserved by synchronous logical transports, e.g., when all slots are reserved and the control logical link (LMP) has data to send.

The ASB logical transport is used by a master to communicate with active slaves. The PSB logical transport is used by a master for communications from the master to parked slave devices. Note that these communications may also be received by active devices.

8.4.2 Logical transport address (LT_ADDR)

Each slave active in a piconet is assigned a primary 3-bit LT_ADDR. The all-zero LT_ADDR is reserved for broadcast messages. The master does not have an LT_ADDR. A master's timing relative to the slaves' distinguishes it from the slaves. A secondary LT_ADDR is assigned to the slave for each eSCO logical transport in use in the piconet. Only eSCO traffic (i.e., NULL, POLL, and one of the **EV** packet types as negotiated at eSCO logical transport setup) may be sent on these LT_ADDRs. ACL traffic (including LMP) shall always be sent on the primary LT_ADDR. A slave shall accept only packets with matching primary or secondary LT_ADDR and broadcast packets. The LT_ADDR is carried in the packet header (see 8.6.4). The LT_ADDR shall be valid only for as long as a slave is in the active mode. As soon as it is disconnected or parked, the slave shall lose all of its LT_ADDRs.

The primary LT_ADDR shall be assigned by the master to the slave when the slave is activated. This is either at connection establishment, at role switch, or when the slave is unparked. At connection establishment and at role switch, the primary LT_ADDR is carried in the FHS payload. When unparking, the primary LT_ADDR is carried in the unpark message.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.4.3 Synchronous logical transports

The first type of synchronous logical transport, the SCO logical transport, is a symmetric, point-to-point link between the master and a specific slave. The SCO logical transport reserves slots and can, therefore, be considered as a circuit-switched connection between the master and the slave. The master may support up to three SCO links to the same slave or to different slaves. A slave may support up to three SCO links from the same master or two SCO links if the links originate from different masters. SCO packets are never retransmitted.

The second type of synchronous logical transport, the eSCO logical transport, is a point-to-point logical transport between the master and a specific slave. eSCO logical transports may be symmetric or asymmetric. Similar to SCO, eSCO reserves slots and can, therefore, be considered a circuit-switched connection between the master and the slave. In addition to the reserved slots, eSCO supports a retransmission window immediately following the reserved slots. Together, the reserved slots and the retransmission window form the complete eSCO window.

8.4.4 Asynchronous logical transport

In the slots not reserved for synchronous logical transports, the master may exchange packets with any slave on a per-slot basis. The ACL logical transport provides a packet-switched connection between the master and all active slaves participating in the piconet. Both asynchronous and isochronous services are supported. Between a master and a slave, only a single ACL logical transport shall exist. For most ACL packets, packet retransmission is applied to assure data integrity.

ACL packets not addressed to a specific slave are considered as broadcast packets and should be read by all slaves that receive them. If there are no data to be sent on the ACL logical transport and no polling is required, no transmission is required.

8.4.5 Transmit/receive routines

This subclause describes the way to use the packets as defined in IEEE 802.15.1-2005 device in order to support the traffic on the ACL, SCO, and eSCO links. Both single-slave and multislave configurations are considered.

8.4.5.1 Flow control

Since the RX ACL buffer can be full while a new payload arrives, flow control is required. The header field FLOW in the return TX packet may use “stop” or “go” in order to control the transmission of new data.

8.4.5.1.1 Destination control

As long as data cannot be received, a stop indication shall be transmitted, and it is automatically inserted by the link controller into the header of the return packet. “Stop” shall be returned as long as the RX ACL buffer is not emptied by the BB resource manager. When new data can be accepted again, the go indication shall be returned. “Go” shall be the default value. All packet types not including data can still be received. Voice communication, for example, is not affected by the flow control. Although a device cannot receive new information, it may still continue to transmit information: the flow control shall be separate for each direction.

8.4.5.1.2 Source control

On the reception of a stop signal, the link controller shall automatically switch to the default packet type. The ACL packet transmitted just before the reception of the stop indication shall be kept until a go signal is received. It may be retransmitted as soon as a go indication is received. Only default packets shall be sent as

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

long as the stop indication is received. When no packet is received, “go” shall be assumed implicitly. Note that the default packets contain link control information (in the header) for the receive direction (which may still be open) and may contain synchronous data (**HV** or **EV** packets). When a go indication is received, the link controller may resume transmitting the data that are present in the TX ACL buffers.

In a multislave configuration, only the transmission to the slave that issued the stop signal shall be stalled. This means that the master shall stop transmission only from the TX ACL buffer corresponding to the slave that momentarily cannot accept data.

8.4.6 Active slave broadcast (ASB) transport

The ASB logical transport is used to transport L2CAP user traffic to all devices in the piconet that are currently connected to the piconet physical channel that is used by the ASB. There is no acknowledgment protocol, and the traffic is unidirectional from the piconet master to the slaves. The ASB logical transport may be used only for L2CAP group traffic and shall never be used for L2CAP connection-oriented channels, L2CAP control signalling, or LMP control signalling.

The ASB logical transport is unreliable. To improve reliability somewhat each packet is transmitted a number of times. An identical SEQN is used to assist with filtering retransmissions at the slave device.

The ASB logical transport is identified by the reserved, all-zero LT_ADDR. Packets on the ASB logical transport may be sent by the master at any time.

8.4.7 Parked slave broadcast (PSB) transport

The PSB logical transport is used for communication from the master to the slaves that are parked. The PSB logical transport is more complex than the other logical transports as it consists of a number of phases, each having a different purpose. These phases are the control information phase (used to carry the LMP logical link), the user information phase (used to carry the L2CAP logical link), and the access phase (carrying BB signalling).

The PSB logical transport is identified by the reserved, all-zero LT_ADDR.

8.4.7.1 Parked member address (PM_ADDR)

A slave in the PARK state can be identified by its BD_ADDR or by a dedicated PM_ADDR. This latter address is an 8-bit member address that separates the parked slaves. The PM_ADDR shall be valid only for as long as the slave is parked. When the slave is activated, it shall be assigned an LT_ADDR, but shall lose the PM_ADDR. The PM_ADDR is assigned to the slave by the master during the parking procedure (see 9.3.5.2).

The all-zero PM_ADDR shall be reserved for parked slaves that use only their BD_ADDR to be unparked.

8.4.7.2 Access request address (AR_ADDR)

The AR_ADDR is used by the parked slave to determine the slave-to-master half slot in the access window where it is allowed to send access request messages (see also 8.8.9.6). The AR_ADDR shall be assigned to the slave when it enters the PARK state and shall be valid only for as long as the slave is parked. The AR_ADDR is not necessarily unique, i.e., different parked slaves may have the same AR_ADDR.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.5 Logical links

Four logical links are defined as follows:

— Link control (LC)
— ACL control (ACL-C)
— Asynchronous/Isochronous user (ACL-U)
— Stream (SCO-S or eSCO-S)

The control logical links LC and ACL-C are used at the link control level and LM level, respectively. The ACL-U logical link is used to carry either asynchronous or isochronous user information. The stream logical link is used to carry synchronous user information. The LC logical link is carried in the packet header; all other logical links are carried in the packet payload. The ACL-C and ACL-U logical links are indicated in the LLID field in the payload header. The SCO-S and eSCO-S logical links are carried by the synchronous logical transports only. The ACL-U link is normally carried by the ACL logical transport; however, they may also be carried by the data in the **DV** packet on the SCO logical transport. The ACL-C link may be carried either by the SCO or the ACL logical transport.

8.5.1 Link control (LC) logical link

The LC logical link shall be mapped onto the packet header. This logical link carries low-level link control information like ARQ, flow control, and payload characterization. The LC logical link is carried in every packet except in the ID packet, which does not have a packet header.

8.5.2 ACL control (ACL-C) logical link

The ACL-C logical link shall carry control information exchanged between the LMs of the master and the slave(s). The ACL-C logical link shall use **DM1** packets. The ACL-C logical link is indicated by the LLID code 11 in the payload header.

8.5.3 Asynchronous/Isochronous user (ACL-U) logical link

The ACL-U logical link shall carry L2CAP asynchronous and isochronous user data. These messages may be transmitted in one or more BB packets. For fragmented messages, the start packet shall use an LLID code of 10 in the payload header. Remaining continuation packets shall use LLID code 01. If there is no fragmentation, all packets shall use the LLID start code 10.

8.5.3.1 Pausing the ACL-U logical link

When the ACL-U logical link is paused by the LM, the link controller transmits the current packet with ACL-U information, if any, until an ACK is received or, optionally, until an explicit negative acknowledge (NAK) is received. While the ACL-U logical link is paused, the link controller shall not transmit any packets with ACL-U logical link information.

If the ACL-U logical link was paused after an ACK, the next SEQN shall be used on the next packet. If the ACL-U logical link was paused after a NAK, the same SEQN shall be used on the next packet, and the unacknowledged packet shall be transmitted once the ACL-U logical link is unpaused.

When the ACL-U logical link is unpaused by the LM, the link controller may resume transmitting packets with ACL-U information.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.5.4 Stream logical link

The stream logical link carries transparent synchronous data. This logical link may be carried over the synchronous logical transport (SCO) or over the extended synchronous logical transport (eSCO).

8.5.5 Logical link priorities

The ACL-C logical link shall have a higher priority than the ACL-U logical link when scheduling traffic on the shared ACL logical transport, except in the case when retransmissions of unacknowledged ACL packets shall be given priority over traffic on the ACL-C logical link. The ACL-C logical link should also have priority over traffic on the stream logical link, but opportunities for interleaving the logical links should be taken.

8.6 Packets

IEEE 802.15.1-2005 devices shall use the packets as defined in 8.6.1 through 8.6.7.

8.6.1 General format

The general packet format is shown in Figure 32. Each packet consists of three entities: the access code, the header, and the payload. In the figure, the number of bits per entity is indicated.

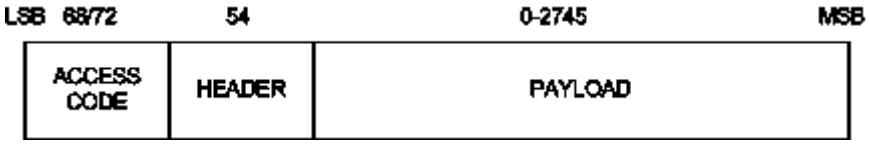


Figure 32—General packet format

The access code is 72 or 68 bits and the header is 54 bits. The payload ranges from zero to a maximum of 2745 bits. Different packet types have been defined. Packet may consist of the following:

— The shortened access code only
— The access code and the packet header
— The access code, the packet header, and the payload

8.6.2 Bit ordering

The bit ordering when defining packets and messages in BB follows the little Endian format. The following rules apply:

– The LSB corresponds to b_0.
– The LSB is the first bit sent over the air.
– In illustrations, the LSB is shown on the left side.

Furthermore, data fields generated internally at BB level, such as the packet header fields and payload header length, shall be transmitted with the LSB first. For instance, a 3-bit parameter X = 3 is sent as follows:

$$b_0 b_1 b_2 = 110$$

over the air where 1 is sent first and 0 is sent last.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.3 Access code

Every packet starts with an access code. If a packet header follows, the access code is 72 bits long; otherwise, the access code is 68 bits long and is known as a shortened access code. The shortened access code does not contain a trailer. This access code is used for synchronization, dc offset compensation, and identification. The access code identifies all packets exchanged on a physical channel: all packets sent in the same physical channel are preceded by the same access code. In the receiver of the device, a sliding correlator correlates against the access code and triggers when a threshold is exceeded. This trigger signal is used to determine the receive timing.

The shortened access code is used in paging, inquiry, and PARK. In this case, the access code itself is used as a signalling message, and neither a header nor a payload is present.

The access code consists of a preamble, a sync word, and possibly a trailer (see Figure 33). For details, see 8.6.3.1.

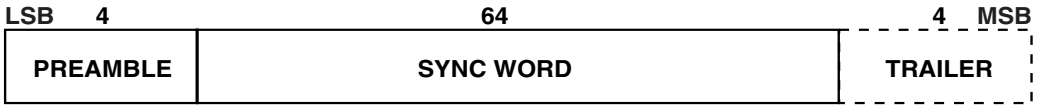


Figure 33—Access code format

8.6.3.1 Access code types

The different access code types use different LAPs to construct the sync word. The LAP field of the BD_ADDR is explained in 8.1.2. A summary of the different access code types is in Table 14.

Table 14—Summary of access code types

Code type	LAP	Code length	Comments
CAC	Master	72	See also 8.1.3
DAC	Paged device	68/72[a]	
GIAC	Reserved	68/72[a]	
DIAC	Dedicated	68/72[a]	

[a]Length 72 is used only in combination with FHS packets.

The CAC consists of a preamble, sync word, and trailer; and its total length is 72 bits. When used as self-contained messages without a header, the DAC and IAC do not include the trailer bits and are 68 bits long.

8.6.3.2 Preamble

The preamble is a fixed zero-one pattern of four symbols used to facilitate dc compensation. The sequence is either 1010 or 0101, depending on whether the LSB of the following sync word is 1 or 0, respectively. The preamble is shown in Figure 34.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Figure 34—Preamble

8.6.3.3 Sync word

The sync word is a 64-bit code word derived from a 24-bit address (LAP). For the CAC, the master's LAP is used; for the GIAC and the DIAC, reserved, dedicated LAPs are used; for the DAC, the slave LAP is used. The construction guarantees large Hamming distance between sync words based on different LAPs. In addition, the good auto-correlation properties of the sync word improve timing acquisition.

8.6.3.3.1 Synchronization word definition

The sync words are based on a (64,30) expurgated block code with an overlay (bitwise XOR) of a 64-bit full-length pseudo-random noise (PN) sequence. The expurgated code guarantees large Hamming distance ($d_{min} = 14$) between sync words based on different addresses. The PN sequence improves the auto-correlation properties of the access code. The following steps describe how the sync word shall be generated:

a) Generate information sequence.
b) XOR this with the "information covering" part of the PN overlay sequence.
c) Generate the code word.
d) XOR the code word with all 64 bits of the PN overlay sequence.

The information sequence is generated by appending 6 bits to the 24-bit LAP (step a). The appended bits are 001101 if the MSB of the LAP equals 0. If the MSB of the LAP is 1, the appended bits are 110010. The LAP MSB together with the appended bits constitute a length-seven Barker sequence. The purpose of including a Barker sequence is to further improve the auto-correlation properties. In step b, the information is prescrambled by XORing it with the bits $p_{34} \ldots p_{63}$ of the PN sequence (defined in 8.6.3.3.2). After generating the code word (step c), the complete PN sequence is XORed to the code word (step d). This step descrambles the information part of the code word. At the same time, the parity bits of the code word are scrambled. Consequently, the original LAP and Barker sequence are ensured a role as a part of the access code sync word, and the cyclic properties of the underlying code are removed. The principle is depicted in Figure 35

In the following discussion, binary sequences will be denoted by their corresponding D-transform (in which D^i represents a delay of i time units). Let $p'(D) = p'_0 + p'_1 D + \ldots + p'_{62} D^{62}$ be the 63-bit PN sequence, where p'_0 is the first bit (LSB) leaving the pseudo-random noise generation (see Figure 36), and p'_{62} is the last bit (MSB). To obtain 64 bits, an extra zero is appended at the end of this sequence (thus, $p'(D)$ is unchanged). For notational convenience, the reciprocal of this extended polynomial, $p(D) = D^{63} p'(1/D)$, will be used in the following discussion. This is the sequence $p'(D)$ in reverse order. The 24-bit LAP of the device address is denoted by $a(D) = a_0 + a_1 D + \ldots + a_{23} D^{23}$ (a_0 is the LSB of the device address).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

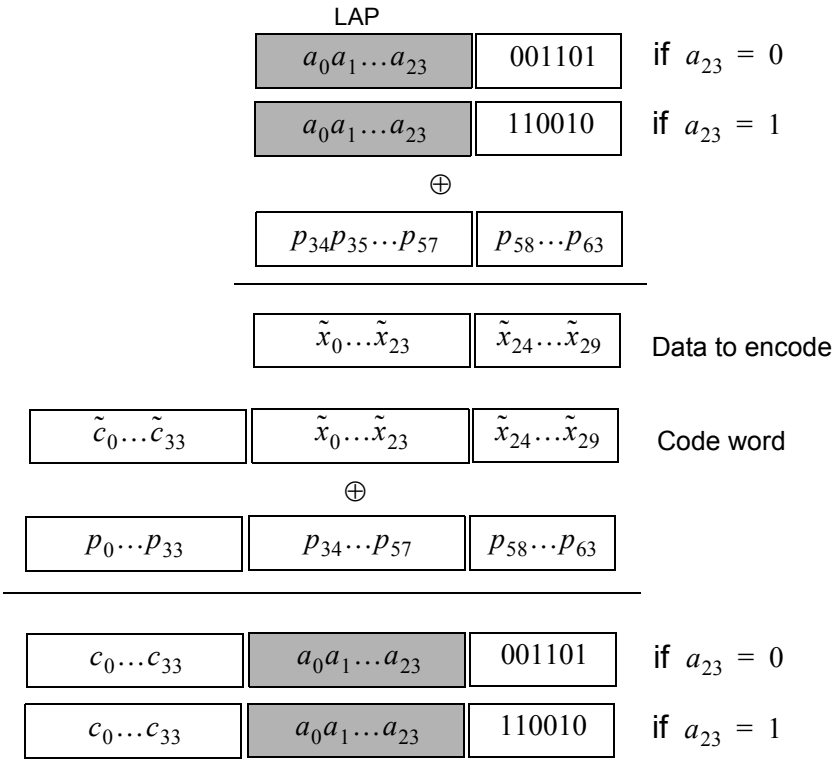


Figure 35—Construction of the sync word

The (64,30) block code generator polynomial is denoted by $g(D) = (1 + D)g'(D)$, where $g'(D)$ is the generator polynomial `157464165547` (octal notation) of a primitive binary (63,30) Bose, Chaudhuri, and Hocquenghem (BCH) code. Thus, in octal notation,

$$g(D) = \texttt{260534236651,} \tag{9}$$

where the leftmost bit corresponds to the high-order (g_{34}) coefficient. The dc-free 4-bit sequences `0101` and `1010` can be written as

$$\begin{cases} F_0(D) = D + D^3, \\ F_1(D) = 1 + D^2, \end{cases} \tag{10}$$

respectively. Furthermore,

$$\begin{cases} B_0(D) = D^2 + D^3 + D^5, \\ B_1(D) = 1 + D + D^4, \end{cases} \tag{11}$$

which are used to create the length seven Barker sequences. Then, the access code shall be generated by the following procedure:

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

a) Format the 30 information bits to encode:

$$x(D) = a(D) + D^{24}B_{a_{23}}(D).$$

b) Add the information covering part of the PN overlay sequence:

$$\tilde{x}(D) = x(D) + p_{34} + p_{35}D + \ldots + p_{63}D^{29}.$$

c) Generate parity bits of the (64,30) expurgated block code:[9]

$$\tilde{c}(D) = D^{34}\tilde{x}(D) \mod g(D).$$

d) Create the code word:

$$\tilde{s}(D) = D^{34}\tilde{x}(D) + \tilde{c}(D).$$

e) Add the PN sequence:

$$s(D) = \tilde{s}(D) + p(D).$$

f) Append the (dc-free) preamble and trailer:

$$y(D) = F_{c_0}(D) + D^4 s(D) + D^{68}F_{a_{23}}(D).$$

8.6.3.3.2 PN sequence generation

To generate the PN sequence, the primitive polynomial $h(D) = 1 + D + D^3 + D^4 + D^6$ shall be used. The linear feedback shift register (LFSR) and its starting state are shown in Figure 36. The PN sequence generated (including the extra terminating zero) becomes (hexadecimal notation) 83848D96BBCC54FC. The LFSR output starts with the leftmost bit of this PN sequence. This corresponds to $p'(D)$ of 8.6.3.3.1. Thus, using the reciprocal $p(D)$ as overlay gives the 64-bit sequence:

$$p = 3F2A33DD69B121C1, \tag{12}$$

where the leftmost bit is $p_0 = 0$ (there are two initial zeros in the binary representation of the hexadecimal digit 3), and $p_{63} = 1$ is the rightmost bit.

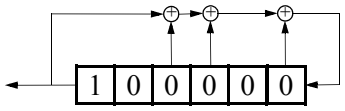


Figure 36—LFSR and the starting state to generate *p′*(*D*)

8.6.3.4 Trailer

The trailer is appended to the sync word as soon as the packet header follows the access code. This is typically the case with the CAC, but the trailer is also used in the DAC and IAC when these codes are used in FHS packets exchanged during page response and inquiry response.

The trailer is a fixed zero-one pattern of four symbols. The trailer together with the 3 MSBs of the sync word form a 7-bit pattern of alternating ones and zeroes, which may be used for extended dc compensation. The trailer sequence is either 1010 or 0101 depending on whether the MSB of the sync word is 0 or 1, respectively. The choice of trailer is illustrated in Figure 37.

[9] $x(D)$ mod $y(D)$ denotes the remainder when $x(D)$ is divided by $y(D)$.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Figure 37—Trailer in CAC when MSB of sync word is 0 (a) and when MSB of sync word is 1 (b)

8.6.4 Packet header

The header contains link control information and consists of six fields:

— LT_ADDR: 3-bit logical transport address
— TYPE: 4-bit type code
— FLOW: 1-bit flow control
— ARQN: 1-bit acknowledge indication
— SEQN: 1-bit sequence number
— HEC: 8-bit header error check

The total header, including the HEC, consists of 18 bits (see Figure 38) and is encoded with a rate 1/3 FEC (not shown, but described in 8.7.4) resulting in a 54-bit header. The LT_ADDR and TYPE fields shall be sent LSB first.



Figure 38—Header format

8.6.4.1 LT_ADDR field

The 3-bit LT_ADDR field contains the LT_ADDR for the packet (see 8.4.2). This field indicates the destination slave for a packet in a master-to-slave transmission slot and indicates the source slave for a slave-to-master transmission slot.

8.6.4.2 TYPE field

Sixteen different types of packets can be distinguished. The 4-bit TYPE code specifies which packet type is used. The interpretation of the TYPE code depends on the LT_ADDR in the packet. First, it shall be determined whether the packet is sent on an SCO logical transport, an eSCO logical transport, or an ACL logical transport. Then it can be determined which type of SCO packet, eSCO packet, or ACL packet has been received. The TYPE code determines how many slots the current packet will occupy (see the "Slot occupancy" column in Table 15 in 8.6.5). This allows the nonaddressed receivers to refrain from listening to the channel for the duration of the remaining slots. In 8.6.5, each packet type is described in more detail.

8.6.4.3 FLOW field

The FLOW bit is used for flow control of packets over the ACL logical transport. When the RX buffer for the ACL logical transport in the recipient is full, a stop indication (FLOW = 0) shall be returned to stop the other device from transmitting data temporarily. The stop signal affects only ACL packets. Packets including only link control information (ID, POLL, and NULL packets), SCO packets, or eSCO packets can still

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

be received. When the RX buffer can accept data, a go indication (FLOW = 1) shall be returned. When no packet is received or the received header is in error, a go indication shall be assumed implicitly. In this case, the slave can receive a new packet with CRC although its RX buffer is still not emptied. The slave shall then return a NAK in response to this packet even if the packet passed the CRC check.

The FLOW bit is not used on the eSCO logical transport or the ACL-C logical link and shall be set to one on transmission and ignored upon receipt.

8.6.4.4 ARQN field

The 1-bit acknowledgment indication ARQN is used to inform the source of a successful transfer of payload data with CRC and can be positive (ACK) or negative (NAK). See 8.7.6 for initialization and usage of this bit.

8.6.4.5 SEQN field

The SEQN bit provides a sequential numbering scheme to order the data packet stream. See 8.7.6.2 for initialization and usage of the SEQN bit. For broadcast packets, a modified sequencing method is used (see 8.7.6.5).

8.6.4.6 HEC field

Each header has an HEC to check the header integrity. The HEC is an 8-bit word (generation of the HEC is specified in 8.7.1.1). Before generating the HEC, the HEC generator is initialized with an 8-bit value. For FHS packets sent in **master response** substate, the slave upper address part (UAP) shall be used. For FHS packets sent in **inquiry response** substate, the DCI (see 8.1.2.1) shall be used. In all other cases, the UAP of the master device shall be used.

After the initialization, a HEC shall be calculated for the 10 header bits. Before checking the HEC, the receiver shall initialize the HEC check circuitry with the proper 8-bit UAP (or DCI). If the HEC does not check, the entire packet shall be discarded. More information can be found in 8.7.1.

8.6.5 Packet types

The packets used on the piconet are related to the logical transports in which they are used. Three logical transports with distinct packet types are defined (see 8.4): the SCO logical transport, the eSCO logical transport, and the ACL logical transport. (The ASB and PSB logical transports do not have distinct packet types, but use the packet types defined for the ACL logical transport.) For each of these logical transports, 15 different packet types can be defined.

To indicate the different packets on a logical transport, the 4-bit TYPE code is used. The packet types are divided into four segments. The first segment is reserved for control packets. All control packets occupy a single time slot. The second segment is reserved for packets occupying a single time slot. The third segment is reserved for packets occupying three time slots. The fourth segment is reserved for packets occupying five time slots. The slot occupancy is reflected in the segmentation and can directly be derived from the type code. Table 15 summarizes the packets defined for the SCO, eSCO, and ACL logical transport types.

8.6.5.1 Common packet types

There are five common kinds of packets. In addition to the types listed in segment 1 of Table 15, the ID packet is also a common packet type. It is not listed in segment 1, however, because it does not have a packet header.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 15—Packets defined for synchronous and asynchronous logical transport types

Segment	TYPE code $b_3b_2b_1b_0$	Slot occupancy	SCO logical transport	eSCO logical transport	ACL logical transport
1	0000	1	NULL	NULL	NULL
	0001	1	POLL	POLL	POLL
	0010	1	FHS	Reserved	FHS
	0011	1	**DM1**	Reserved	**DM1**
2	0100	1	Undefined	Undefined	**DH1**
	0101	1	**HV1**	Undefined	Undefined
	0110	1	**HV2**	Undefined	Undefined
	0111	1	**HV3**	**EV3**	Undefined
	1000	1	**DV**	Undefined	Undefined
	1001	1	Undefined	Undefined	**AUX1**
3	1010	3	Undefined	Undefined	**DM3**
	1011	3	Undefined	Undefined	**DH3**
	1100	3	Undefined	**EV4**	Undefined
	1101	3	Undefined	**EV5**	Undefined
4	1110	5	Undefined	Undefined	**DM5**
	1111	5	Undefined	Undefined	**DH5**

8.6.5.1.1 ID packet

The ID packet consists of the DAC or IAC. It has a fixed length of 68 bits. It is a very robust packet since the receiver uses a bit correlator to match the received packet to the known bit sequence of the ID packet.

8.6.5.1.2 NULL packet

The NULL packet has no payload and consists of the CAC and packet header only. Its total (fixed) length is 126 bits. The NULL packet may be used to return link information to the source regarding the success of the previous transmission (ARQN) or the status of the RX buffer (FLOW). The NULL packet does not require an acknowledgment.

8.6.5.1.3 POLL packet

The POLL packet is very similar to the NULL packet. It does not have a payload. In contrast to the NULL packet, it requires a confirmation from the recipient. It is not a part of the ARQ scheme. The POLL packet does not affect the ARQN and SEQN fields. Upon reception of a POLL packet, the slave shall respond with a packet even when the slave does not have any information to send unless the slave has scatternet commitments in that timeslot. This return packet is an implicit acknowledgment of the POLL packet. This packet can be used by the master in a piconet to poll the slaves. Slaves shall not transmit the POLL packet.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.5.1.4 FHS packet

The FHS packet is a special control packet containing, among other things, the device address and the clock of the sender. The payload contains 144 information bits plus a 16-bit CRC code. The payload is coded with a rate 2/3 FEC with a gross payload length of 240 bits.

Figure 39 illustrates the format and contents of the FHS payload. The payload consists of 11 fields. The FHS packet is used in page master response, inquiry response and role switch.

The FHS packet contains real-time clock information. This clock information shall be updated before each retransmission. The retransmission of the FHS payload is different from retransmissions of ordinary data payloads where the same payload is used for each retransmission. The FHS packet is used for frequency hop synchronization before the piconet physical channel has been established or when an existing piconet changes to a new piconet.

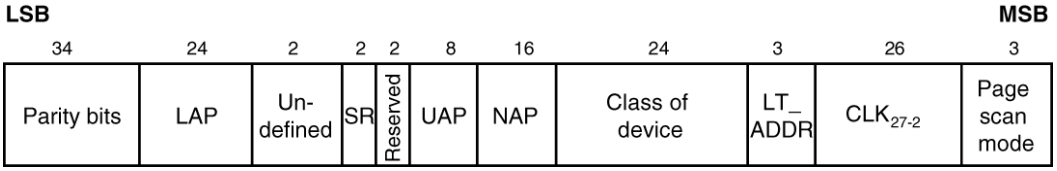


Figure 39—Format of the FHS payload

Each field is described in more detail below and in Table 16:

Table 16—Description of the FHS payload

Field name	Description
Parity Bits	This 34-bit field contains the parity bits that form the first part of the sync word of the access code of the device that sends the FHS packet. These bits are derived from the LAP as described in 8.1.2.
LAP	This 24-bit field shall contain the LAP of the device that sends the FHS packet.
Undefined	This 2-bit field is reserved for future use and shall be set to zero.
SR	This 2-bit field is the scan repetition field and indicates the interval between two consecutive page scan windows (see also Table 17 and Table 24).
Reserved	This 2-bit field shall be set to 10.
UAP	This 8-bit field shall contain the UAP of the device that sends the FHS packet.
NAP	This 16-bit field shall contain the NAP of the device that sends the FHS packet (see also 8.1.2 for LAP, UAP, and NAP).
Class of Device	This 24-bit field shall contain the class of the device that sends the FHS packet. The field is defined in Bluetooth Assigned Numbers [B1].
LT_ADDR	This 3-bit field shall contain the LT_ADDR the recipient shall use if the FHS packet is used at connection setup or role switch. A slave responding to a master or a device responding to an inquiry request message shall include an all-zero LT_ADDR field if it sends the FHS packet.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 16—Description of the FHS payload *(continued)*

Field name	Description
CLK_{27-2}	This 26-bit field shall contain the value of the CLKN of the device that sends the FHS packet, sampled at the beginning of the transmission of the access code of this FHS packet. This clock value has a resolution of 1.25 ms (two-slot interval). For every new transmission, this field is updated so that it accurately reflects the real-time clock value.
Page Scan Mode	This 3-bit field shall indicate which scan mode is used by the sender of the FHS packet. The interpretation of the page scan mode is illustrated in Table 18.

— The device sending the FHS shall set the SR bits according to Table 17.

Table 17—Contents of SR field

SR bit format b_1b_0	SR mode
00	R0
01	R1
10	R2
11	Reserved

— The device sending the FHS shall set the Page Scan Mode bits according to Table 18.

Table 18—Contents of Page Scan Mode field

Bit format $b_2b_1b_0$	Page scan mode
000	Mandatory scan mode
001	Reserved for future use
010	Reserved for future use
011	Reserved for future use
100	Reserved for future use
101	Reserved for future use
110	Reserved for future use
111	Reserved for future use

— The LAP, UAP, and NAP together form the 48-bit address of the device that sends the FHS packet. Using the parity bits and the LAP, the recipient can directly construct the CAC of the sender of the FHS packet.
— The DCI shall be used to initialize the HEC and CRC for the FHS packet of the inquiry response (see 8.1.2.1).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.5.1.5 DM1 packet

DM1 is part of segment 1 in order to support control messages in any logical transport that allows the **DM1** packet (see Table 15). However, it may also carry regular user data. Since the **DM1** packet can be regarded as an ACL packet, it will be discussed in 8.6.5.4.

8.6.5.2 SCO packets

HV and **DV** packets are used on the synchronous SCO logical transport. The **HV** packets do not include a CRC and shall not be retransmitted. **DV** packets include a CRC on the data portion, but not on the synchronous data portion. The data portion of **DV** packets shall be retransmitted if it is not acknowledged. SCO packets may be routed to the synchronous I/O port. Four packets are allowed on the SCO logical transport: **HV1**, **HV2**, **HV3**, and **DV**. These packets are typically used for 64 kb/s speech transmission, but may be used for transparent synchronous data.

8.6.5.2.1 HV1 packet

The **HV1** packet has 10 information bytes. The bytes are protected with a rate 1/3 FEC. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

8.6.5.2.2 HV2 packet

The **HV2** packet has 20 information bytes. The bytes are protected with a rate 2/3 FEC. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

8.6.5.2.3 HV3 packet

The **HV3** packet has 30 information bytes. The bytes are not protected by FEC. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

8.6.5.2.4 DV packet

The **DV** packet is a combined data-voice packet. The **DV** packet shall be used only in place of an **HV1** packet. The payload is divided into a Voice field of 80 bits and a Data field containing up to 150 bits (see Figure 40). The Voice field is not protected by FEC. The Data field has between 1 and 10 information bytes (including the 1-byte payload header) and includes a 16-bit CRC. The data field is encoded with a rate 2/3 FEC. Since the **DV** packet has to be sent at regular intervals due to its synchronous contents, it is listed under the SCO packet types. The Voice and Data fields shall be treated separately. The Voice field shall be handled in the same way as normal SCO data and shall never be retransmitted, i.e., the Voice field is always new. The Data field is checked for errors and shall be retransmitted if necessary. When the asynchronous data field in the **DV** packet has not been acknowledged before the SCO logical transport is terminated, the asynchronous Data field shall be retransmitted in a **DM1** packet.

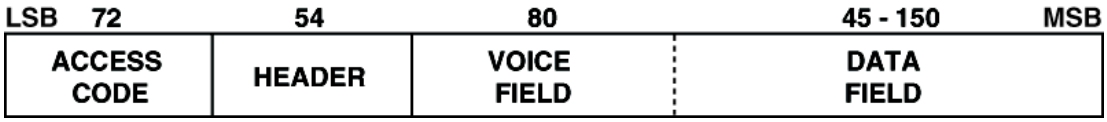


Figure 40—DV packet format

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.5.3 eSCO packets

EV packets are used on the synchronous eSCO logical transport. The packets include a CRC and retransmission may be applied if no acknowledgment of proper reception is received within the retransmission window. eSCO packets may be routed to the synchronous I/O port. Three eSCO packets have been defined. The eSCO packets may be used for 64 kb/s speech transmission as well as transparent data at 64 kb/s and other rates.

8.6.5.3.1 EV3 packet

The **EV3** packet has between 1 and 30 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The **EV3** packet may cover up to a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or renegotiated.

8.6.5.3.2 EV4 packet

The **EV4** packet has between 1 and 120 information bytes plus a 16-bit CRC code. The **EV4** packet may cover to up three time slots. The information plus CRC bits are coded with a rate 2/3 FEC. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or renegotiated.

8.6.5.3.3 EV5 packet

The **EV5** packet has between 1 and 180 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The **EV5** packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or renegotiated.

8.6.5.4 ACL packets

ACL packets are used on the asynchronous logical transport. The information carried may be user data or control data.

8.6.5.4.1 DM1 packet

The **DM1** packet carries data information only. The payload has between 1 and 18 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. The **DM1** packet occupies a single time slot. The information plus CRC bits are coded with a rate 2/3 FEC. The payload header in the **DM1** packet is 1 byte long (see Figure 41). The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

8.6.5.4.2 DH1 packet

This packet is similar to the **DM1** packet, except that the information in the payload is not FEC encoded. As a result, the **DH1** packet has between 1 and 28 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. The **DH1** packet occupies a single time slot.

8.6.5.4.3 DM3 packet

The **DM3** packet may occupy up to three time slots. The payload has between 2 and 123 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The information plus CRC bits are coded with a rate 2/3 FEC. The payload header in the **DM3** packet is 2 bytes long (see Figure 42). The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.5.4.4 DH3 packet

This packet is similar to the **DM3** packet, except that the information in the payload is not FEC encoded. As a result, the **DH3** packet has between 2 and 185 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The **DH3** packet may occupy up to three time slots.

8.6.5.4.5 DM5 packet

The **DM5** packet may occupy up to five time slots. The payload has between 2 and 226 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The payload header in the **DM5** packet is 2 bytes long. The information plus CRC bits are coded with a rate 2/3 FEC. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

8.6.5.4.6 DH5 packet

This packet is similar to the **DM5** packet, except that the information in the payload is not FEC encoded. As a result, the **DH5** packet has between 2 and 341 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The **DH5** packet may occupy up to five time slots.

8.6.5.4.7 AUX1 packet

This packet resembles a **DH1** packet, but has no CRC code. The **AUX1** packet has between 1 and 30 information bytes (including the 1-byte payload header). The **AUX1** packet occupies a single time slot. The **AUX1** packet shall not be used for the ACL-U or ACL-C logical links. An **AUX1** packet may be discarded.

8.6.6 Payload format

In the payload, two fields are distinguished: the Synchronous Data field and the Asynchronous Data field. The ACL packets have only the Asynchronous Data field, and the SCO and eSCO packets have only the Synchronous Data field—with the exception of the **DV** packets, which have both.

8.6.6.1 Synchronous Data field

In SCO, the Synchronous Data field has a fixed length and consists only of the synchronous data body portion. No payload header is present.

In eSCO, the Synchronous Data field consists of two segments: a synchronous data body and a CRC code. No payload header is present.

8.6.6.1.1 Synchronous data body

For **HV** and **DV** packets, the synchronous data body length is fixed. For **EV** packets, the synchronous data body length is negotiated during the LMP eSCO setup. Once negotiated, the synchronous data body length remains constant unless renegotiated. The synchronous data body length may be different for each direction of the eSCO logical transport.

8.6.6.1.2 CRC code

The 16-bit CRC in the payload is generated as specified in 8.7.1. The 8-bit UAP of the master is used to initialize the CRC generator.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.6.6.2 Asynchronous data field

ACL packets have an asynchronous data field consisting of two or three segments: a payload header, a payload body, and possibly a CRC code (the **AUX1** packet does not carry a CRC code).

8.6.6.2.1 Payload header

The payload header is one or two bytes long. Packets in segments 1 and 2 (see Table 15) have a 1-byte payload header; packets in segments 3 and 4 (see Table 15) have a 2-byte payload header. The payload header specifies the logical link (2-bit LLID indication), controls the flow on the logical channels (1-bit FLOW indication), and has a payload length indicator (5 bits and 9 bits for 1-byte and 2-byte payload headers, respectively). In the case of a 2-byte payload header, the length indicator is extended by 4 bits into the next byte. The remaining 4 bits of the second byte are reserved for future use and shall be set to zero. The formats of the 1-byte and 2-byte payload headers are shown in Figure 41 and Figure 42.

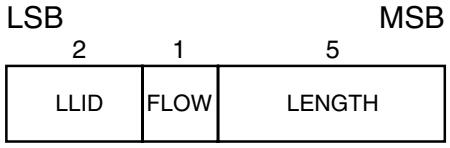


Figure 41—Payload header format for single-slot ACL packets

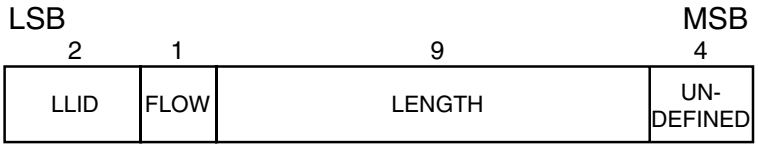


Figure 42—Payload header format for multislot ACL packets

The LLID field shall be transmitted first, the Length field last. In Table 19, more details about the contents of the LLID field are listed.

Table 19—LLID field contents

LLID code b_1b_0	Logical link	Information
00	NA	Undefined
01	ACL-U	Continuation fragment of an L2CAP message
10	ACL-U	Start of an L2CAP message or no fragmentation
11	ACL-C	LMP message

An L2CAP message may be fragmented into several packets. Code 10 shall be used for an ACL-U packet carrying the first fragment of such a message; code 01 shall be used for continuing fragments. If there is no fragmentation, code 10 shall be used for every packet. Code 11 shall be used for LMP messages. Code 00 is reserved for future use.

The flow indicator in the payload is used to control the flow at the L2CAP level. It is used to control the flow per logical link. FLOW = 1 means flow-on (go), and FLOW = 0 means flow-off (stop). After a new

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

connection has been established, the flow indicator shall be set to "go." When a device receives a payload header with the FLOW bit set to "stop," it shall stop the transmission of ACL packets before an additional amount of payload data is sent. This amount is defined as the *flow control lag*, expressed as a number of bytes. The shorter the flow control lag, the less buffering the other device must dedicate to this function. The flow control lag shall not exceed 1792 bytes (7 × 256 bytes). In order to allow devices to optimize the selection of packet length and buffer space, the flow control lag of a given implementation shall be provided in the LMP_features_res message.

If a packet containing the payload FLOW bit of "stop" is received with a valid packet header, but bad payload, the payload flow control bit shall be ignored. The BB ACK contained in the packet header will be received, and a further ACL packet may be transmitted. Each occurrence of this situation allows a further ACL packet to be sent in spite of the flow control request being sent via the payload header flow control bit. It is recommended that devices that use the payload header FLOW bit should ensure that no further ACL packets are sent until the payload FLOW bit has been correctly received. This can be accomplished by simultaneously turning on the FLOW bit in the packet header and keeping it on until an ACK is received back (ARQN = 1). This will typically be only one round-trip time. Since they lack a payload CRC, **AUX1** packets should not be used with a payload FLOW bit of "stop."

The BB resource manager is responsible for setting and processing the FLOW bit in the payload header. Real-time flow control shall be carried out at the packet level by the link controller via the FLOW bit in the packet header (see 8.6.4.3). With the payload FLOW bit, traffic from the remote end can be controlled. It is allowed to generate and send an ACL packet with payload length zero irrespective of flow status. L2CAP start-fragment and continue-fragment indications (LLID = 10 and LLID = 01) also retain their meaning when the payload length is equal to zero (i.e., an empty start-fragment shall not be sent in the middle of an on-going ACL-U packet transmission). It is always safe to send an ACL packet with length = 0 and LLID = 01. The payload FLOW bit has its own meaning for each logical link (ACL-U or ACL-C) (see Table 20). On the ACL-C logical link, no flow control is applied, and the payload FLOW bit shall always be set to one.

Table 20—Use of payload header FLOW bit on the logical links

LLID code b_1b_0	Usage and semantics of the ACL payload header FLOW bit
00	Not defined, reserved for future use
01 or 10	Flow control of the ACL-U channel (L2CAP messages)
11	Always set FLOW = 1 on transmission and ignore the bit on reception

The length indicator shall be set to the number of bytes (i.e., 8-bit words) in the payload excluding the payload header and the CRC code, i.e., the payload body only. With reference to Figure 41 and Figure 42, the MSB of the length field in a 1-byte header is the last (rightmost) bit in the payload header; the MSB of the length field in a 2-byte header is the fourth bit (from left) of the second byte in the payload header.

8.6.6.2.2 Payload body

The payload body includes the user information and determines the effective user throughput. The length of the payload body is indicated in the length field of the payload header.

8.6.6.2.3 CRC code generation

The 16-bit CRC code in the payload is generated as specified in 8.7.1. Before determining the CRC code, an 8-bit value is used to initialize the CRC generator. For the CRC code in the FHS packets sent in **master**

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

response substate, the UAP of the slave is used. For the FHS packet sent in **inquiry response** substate, the DCI (see 8.1.2.1) is used. For all other packets, the UAP of the master is used.

8.6.7 Packet summary

A summary of the packets and their characteristics are shown in Table 21, Table 22, and Table 23. The payload represents the packet payload excluding FEC, CRC, and payload header.

Table 21—Link control packets

Type	Payload (bytes)	FEC	CRC	Symmetric maximum rate	Asymmetric maximum rate
ID	NA	NA	NA	NA	NA
NULL	NA	NA	NA	NA	NA
POLL	NA	NA	NA	NA	NA
FHS	18	2/3	Yes	NA	NA

Table 22—ACL packets

Type	Payload header (bytes)	Payload (bytes)	FEC	CRC	Symmetric maximum rate (kb/s)	Asymmetric maximum rate (kb/s)	
						Forward	Reverse
DM1	1	0–17	2/3	Yes	108.8	108.8	108.8
DH1	1	0–27	No	Yes	172.8	172.8	172.8
DM3	2	0–121	2/3	Yes	258.1	387.2	54.4
DH3	2	0–183	No	Yes	390.4	585.6	86.4
DM5	2	0–224	2/3	Yes	286.7	477.8	36.3
DH5	2	0–339	No	Yes	433.9	723.2	57.6
AUX1	1	0–29	No	No	185.6	185.6	185.6

Table 23—Synchronous packets

Type	Payload header (bytes)	Payload (bytes)	FEC	CRC	Symmetric maximum rate (kb/s)
HV1	NA	10	1/3	No	64.0
HV2	NA	20	2/3	No	64.0
HV3	NA	30	No	No	64.0
DV[a]	1 D	10 + (0 – 9) D	2/3 D	Yes D	64.0 + 57.6 D

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 23—Synchronous packets *(continued)*

Type	Payload header (bytes)	Payload (bytes)	FEC	CRC	Symmetric maximum rate (kb/s)
EV3	NA	1-30	No	Yes	96
EV4	NA	1–120	2/3	Yes	192
EV5	NA	1–180	No	Yes	288

[a]Items followed by D relate to data field only.

8.7 Bitstream processing

IEEE 802.15.1-2005 devices shall use the bitstream processing schemes as defined in 8.7.1 through 8.7.6.

Before the payload is sent over the air interface, several bit manipulations are performed in the transmitter to increase reliability and security. An HEC is added to the packet header, the header bits are scrambled with a whitening word, and FEC coding is applied. In the receiver, the inverse processes are carried out. Figure 43 shows the processes carried out for the packet header both at the transmit and the receive side. All header bit processes are mandatory.

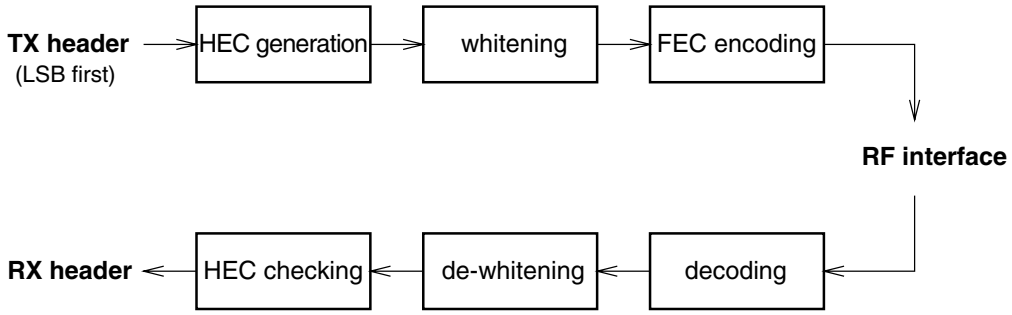


Figure 43—Header bit processes

Figure 44 shows the processes that may be carried out on the payload. In addition to the processes defined for the packet header, encryption may be applied on the payload. Only whitening and dewhitening, as explained in 8.7.2, are mandatory for every payload (with the exception of test mode); all other processes are optional and depend on the packet type (see 8.6.6) and whether encryption is enabled. In Figure 44, optional processes are indicated by dashed blocks.

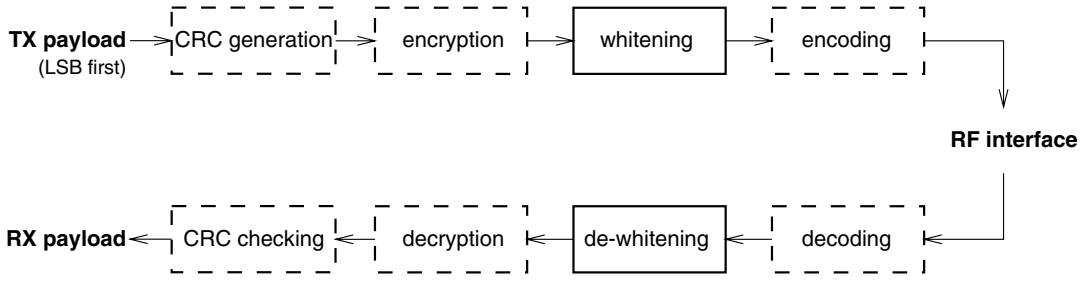


Figure 44—Payload bit processes

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.7.1 Error checking

The packet can be checked for errors or wrong delivery using the CAC, the HEC in the header, and the CRC in the payload. At packet reception, the access code is checked first. Since the 64-bit sync word in the CAC is derived from the 24-bit master LAP, this checks if the LAP is correct and prevents the receiver from accepting a packet of another piconet (provided the LAP field of the master's BD_ADDR is different).

The HEC and CRC computations are normally initialized with the UAP of the master. Even though the access code may be the same for two piconets, the different UAP values will typically cause the HEC and CRC to fail. However, there is an exception where no common UAP is available in the transmitter and receiver. This is the case when the HEC and CRC are generated for the FHS packet in **inquiry response** substate. In this case the DCI value shall be used.

The generation and check of the HEC and CRC are summarized in Figure 47 and Figure 50. Before calculating the HEC or CRC, the shift registers in the HEC/CRC generators shall be initialized with the 8-bit UAP (or DCI) value. Then the header and payload information shall be shifted into the HEC and CRC generators, respectively (with the LSB first).

8.7.1.1 HEC generation

The HEC generating LFSR is depicted in Figure 45. The generator polynomial is $g(D) = (D + 1)(D^7 + D^4 + D^3 + D^2 + 1) = D^8 + D^7 + D^5 + D^2 + D + 1$. Initially, this circuit shall be preloaded with the 8-bit UAP so that the LSB of the UAP (denoted UAP_0) goes to the leftmost shift register element, and UAP_7 goes to the rightmost element. The initial state of the HEC LFSR is depicted in Figure 46. Then the data shall be shifted in with the switch S set in position 1. When the last data bit has been clocked into the LFSR, the switch S shall be set in position 2, and the HEC can be read out from the register. The LFSR bits shall be read out from right to left (i.e., the bit in position 7 is the first to be transmitted, followed by the bit in position 6, and so on).

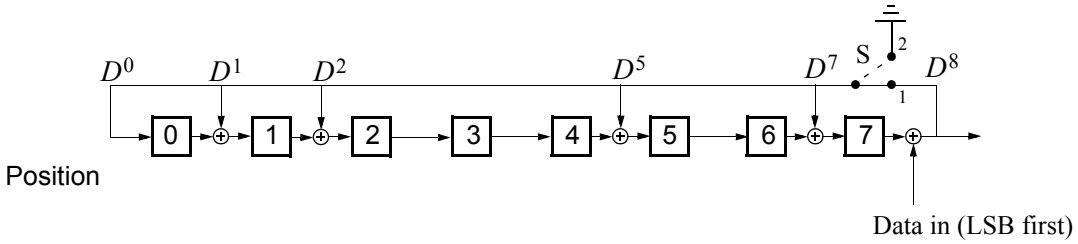


Figure 45—The LFSR circuit generating the HEC

Position:	0	1	2	3	4	5	6	7
LFSR:	UAP_0	UAP_1	UAP_2	UAP_3	UAP_4	UAP_5	UAP_6	UAP_7

Figure 46—Initial state of the HEC generating circuit

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

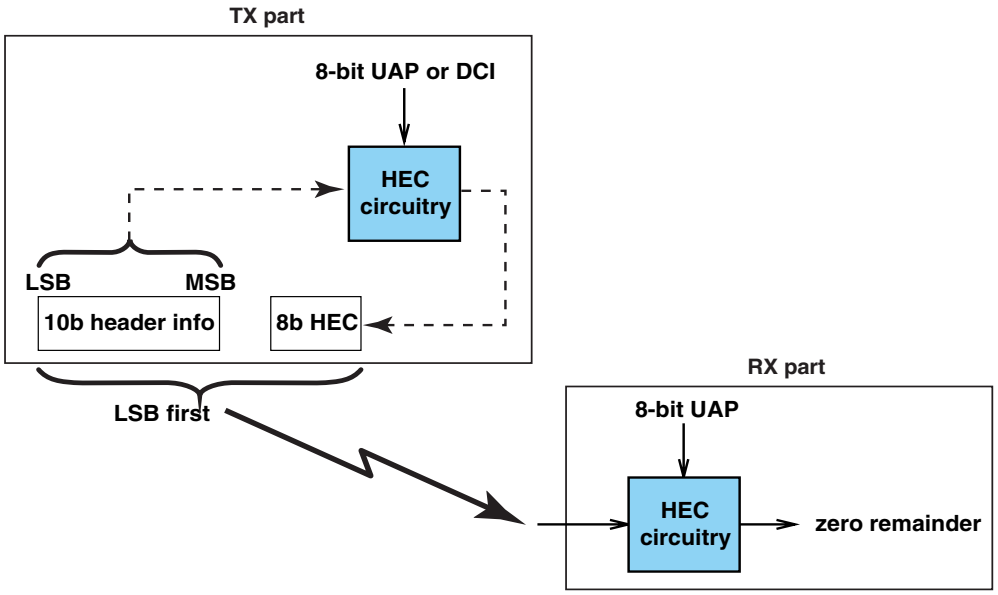


Figure 47—HEC generation and checking

8.7.1.2 CRC generation

The 16-bit LFSR for the CRC is constructed similarly to the HEC using the CRC-CCITT generator polynomial $g(D) = D^{16} + D^{12} + D^5 + 1$ (i.e., `210041` in octal representation) (see Figure 48). For this case, the 8 leftmost bits shall be initially loaded with the 8-bit UAP (UAP_0 to the left and UAP_7 to the right) while the 8 rightmost bits shall be reset to zero. The initial state of the 16-bit LFSR is specified in Figure 49. The switch S shall be set in position 1 while the data are shifted in. After the last bit has entered the LFSR, the switch shall be set in position 2, and the register's contents shall be transmitted, from right to left (i.e., starting with position 15, then position 14, and so on).

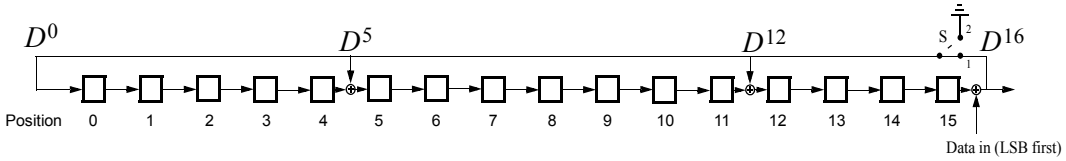


Figure 48—The LFSR circuit generating the CRC

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LFSR:	UAP_0	UAP_1	UAP_2	UAP_3	UAP_4	UAP_5	UAP_6	UAP_7	0	0	0	0	0	0	0	0

Figure 49—Initial state of the CRC generating circuit

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

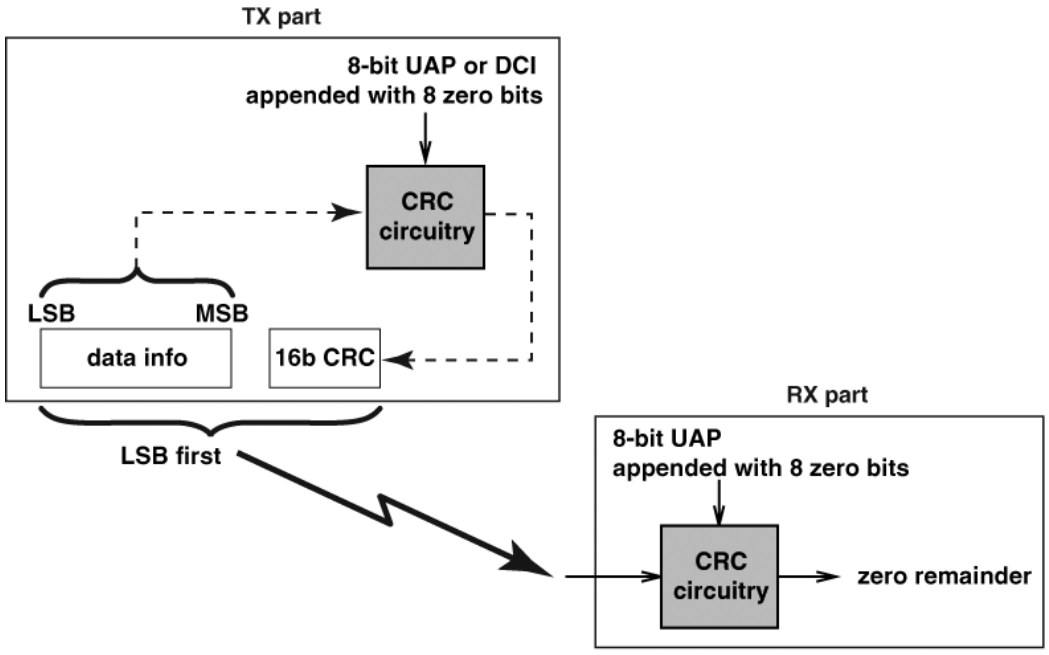


Figure 50—CRC generation and checking

8.7.2 Data whitening

Before transmission, both the header and the payload shall be scrambled with a data whitening word in order to randomize the data from highly redundant patterns and to minimize dc bias in the packet. The scrambling shall be performed prior to the FEC encoding.

At the receiver, the received data shall be descrambled using the same whitening word generated in the recipient. The descrambling shall be performed after FEC decoding.

The whitening word is generated with the polynomial $g(D) = D^7 + D^4 + 1$ (i.e., 221 in octal representation) and shall be subsequently XORed with the header and the payload. The whitening word is generated with the LFSR shown in Figure 51. Before each transmission, the shift register shall be initialized with a portion of the master clock CLK_{6-1}, extended with an MSB of value one. This initialization shall be carried out with CLK_1 written to position 0, CLK_2 written to position 1, etc. An exception is the FHS packet sent during page response or inquiry, where initialization of the whitening register shall be carried out differently. Instead of CLK, the input X used in the inquiry or page response (depending on current state) routine shall be used (see Table 13). The 5-bit value shall be extended with 2 MSBs of value 1. During register initialization, the LSB of X (i.e., X_0) shall be written to position 0, X_1 shall be written to position 1, etc.

After initialization, the packet header and the payload (including the CRC) are whitened. The payload whitening shall continue from the state the whitening LFSR had at the end of HEC. There shall be no reinitialization of the shift register between packet header and payload. The first bit of the “data in” sequence shall be the LSB of the packet header.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

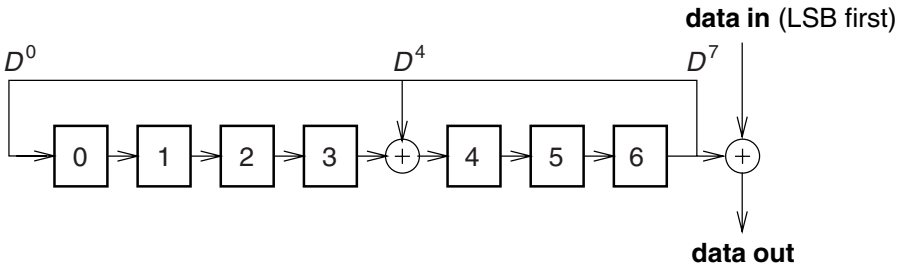


Figure 51—Data whitening LFSR

8.7.3 Error correction

There are three error correction schemes defined for this standard:

— 1/3 rate FEC
— 2/3 rate FEC
— ARQ scheme for the data

The purpose of the FEC scheme on the data payload is to reduce the number of retransmissions. However, in a reasonable error-free environment, FEC gives unnecessary overhead that reduces the throughput. Therefore, the packet definitions given in 8.6 have been kept flexible to use FEC in the payload or not, resulting in the **DM** and **DH** packets for the ACL logical transport, **HV** packets for the SCO logical transport, and **EV** packets for the eSCO logical transport. The packet header is always protected by a 1/3 rate FEC since it contains valuable link information and is designed to withstand more bit errors.

Correction measures to mask errors in the voice decoder are not included in this subclause. This matter is discussed in 8.9.3.

8.7.4 FEC code: rate 1/3

A simple 3-times repetition FEC code is used for the header. The repetition code is implemented by repeating each bit three times (see the illustration in Figure 52). The 3-times repetition code is used for the entire header as well as for the Synchronous Data field in the **HV1** packet.

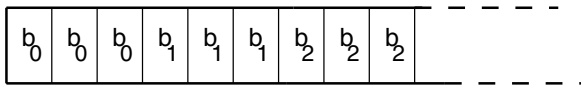


Figure 52—Bit-repetition encoding scheme

8.7.5 FEC code: rate 2/3

The other FEC scheme is a (15,10) shortened Hamming code. The generator polynomial is $g(D) = (D + 1)(D^4 + D + 1)$. This corresponds to 65 in octal notation. An LFSR that may be used to generate this code is depicted in Figure 53. Initially all register elements are set to zero. The 10 information bits are sequentially fed into the LFSR with the switches S1 and S2 set in position 1. Then, after the final input bit, the switches S1 and S2 are set in position 2, and the 5 parity bits are shifted out. The parity bits are appended to the information bits. Subsequently, each block of 10 information bits is encoded into a 15-bit code word. This code can correct all single errors and detect all double errors in each code word. This 2/3 rate FEC is used in the **DM** packets, in the Data field of the **DV** packet, in the FHS packet, in the **HV2** packet, and in the **EV4** packet. Since the encoder operates with information segments of length 10, tail bits with value zero shall be appended after the CRC bits to bring the total number of bits equal to a multiple of 10.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The number of tail bits to append shall be the least possible that achieves this (i.e., in the interval 0...9). These tail bits are not included in the payload length indicator for ACL packets or in the payload length field of the eSCO setup LMP command.

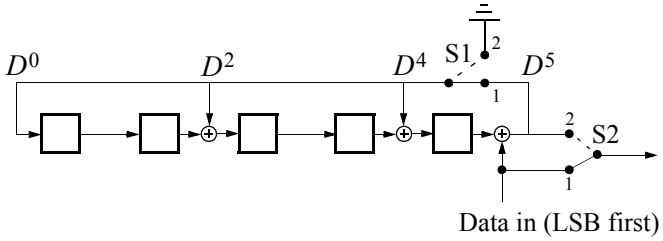


Figure 53—LFSR generating the (15,10) shortened Hamming code

8.7.6 Automatic repeat request (ARQ) scheme

With an ARQ scheme, **DM**, **DH,** the Data field of **DV** packets, and **EV** packets shall be transmitted until acknowledgment of a successful reception is returned by the destination (except when a timeout is exceeded or a flush command is received). The acknowledgment information shall be included in the header of the return packet. The ARQ scheme is used only on the payload in the packet and only on packets that have a CRC. The packet header and the synchronous data payload of **HV** and **DV** packets are not protected by the ARQ scheme.

The rules in 8.7.6.1 through 8.7.6.4 do not apply to broadcast packets.

8.7.6.1 Unnumbered ARQ

IEEE 802.15.1-2005 uses a fast, unnumbered acknowledgment scheme. An ACK (ARQN = 1) or a NAK (ARQN = 0) is returned in response to the receipt of previously received packet. The slave shall respond in the slave-to-master slot directly following the master-to-slave slot unless the slave has scatternet commitments in that timeslot. The master shall respond at the next event addressing the same slave (the master may have addressed other slaves between the last received packet from the considered slave and the master response to this packet). For a packet reception to be successful, at least the HEC must pass. In addition, the CRC must pass if present.

In the first POLL packet at the start of a new connection (as a result of a page, page scan, role switch, or unpark), the master shall initialize the ARQN bit to NAK. The response packet sent by the slave shall also have the ARQN bit set to "NAK." The subsequent packets shall use the following rules. The initial value of the master's eSCO ARQN at link set-up shall be NAK.

The ARQ bit shall be affected only by empty slots and data packets containing CRC. As shown in Figure 54, upon successful reception of a CRC packet, the ARQN bit shall be set to ACK. The ARQN bit shall be set to NAK if, in any receive slot in the slave or in a receive slot in the master following transmission of a packet, one or more of the following events applies:

— No access code is detected.
— The HEC fails.
— The CRC fails.

In eSCO, the receiving device may be able to use an erroneous packet, in which case it may set the ARQN bit to ACK even when the CRC on an **EV** packet has failed. This may be useful to save bandwidth or to save power.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Packets that have correct HEC, but that are addressed to other slaves, or packets other than **DH**, **DM**, **DV**, or **EV** packets shall not affect the ARQN bit, except as noted in 8.7.6.2.2. In these cases, the ARQN bit shall be left as it was prior to reception of the packet. For ACL packets, if a CRC packet with a correct header has the same SEQN as the previously received CRC packet, the ARQN bit shall be set to ACK, and the payload shall be ignored without checking the CRC. For eSCO packets, the SEQN shall not be used when determining the ARQN. If an eSCO packet has been received successfully within the eSCO window, subsequent receptions within the eSCO window shall be ignored. At the end of the eSCO window, the master's ARQN shall be retained for the first master-to-slave transmission in the next eSCO window.

The ARQ bit in the FHS packet is not meaningful. Contents of the ARQN bit in the FHS packet shall not be checked.

Broadcast packets shall be checked on errors using the CRC, but no ARQ scheme shall be applied. Broadcast packets shall never be acknowledged.

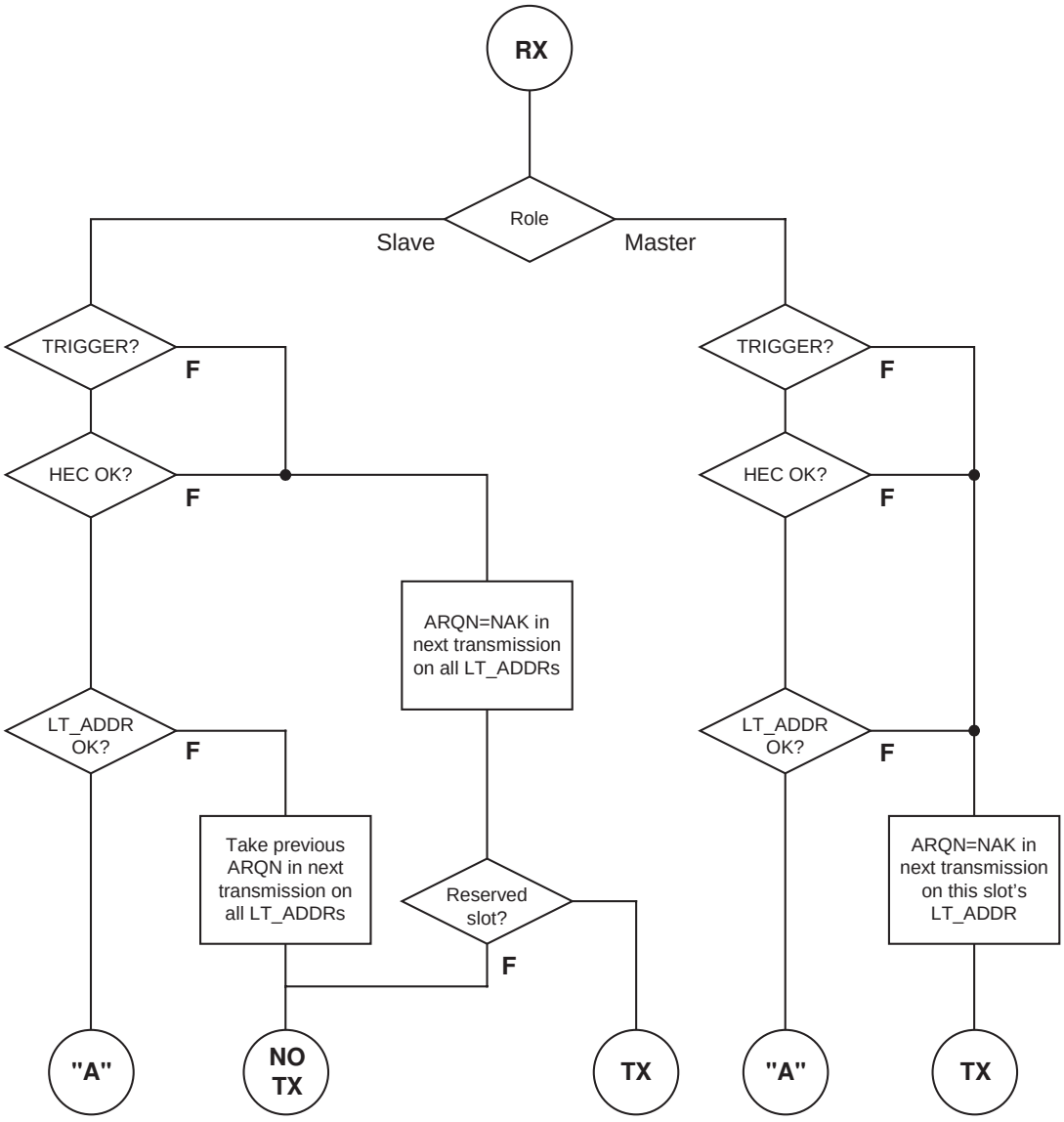


Figure 54—Stage 1 of the receive protocol for determining the ARQN bit

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

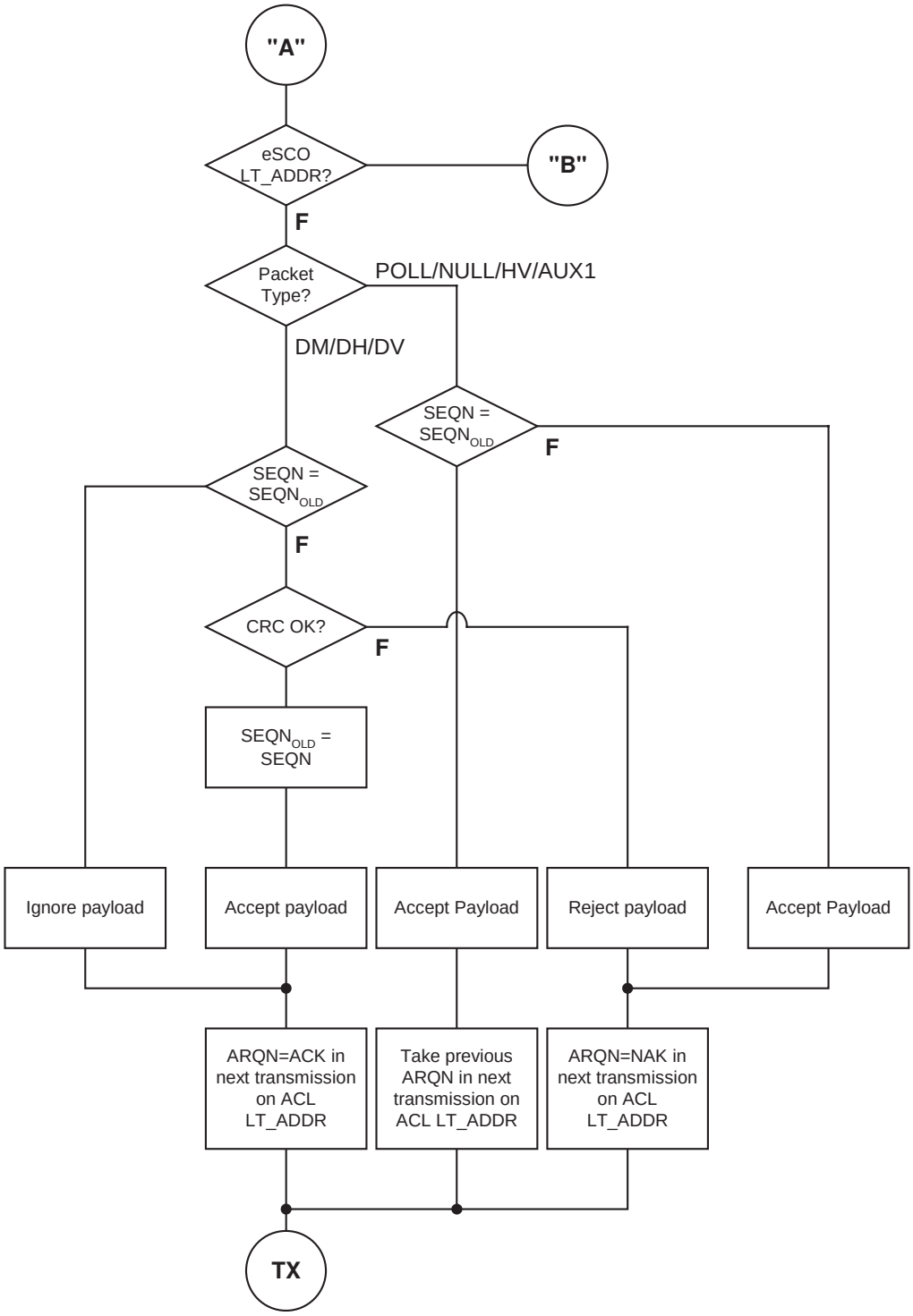


Figure 55—Stage 2 (ACL) of the receive protocol for determining the ARQN bit

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

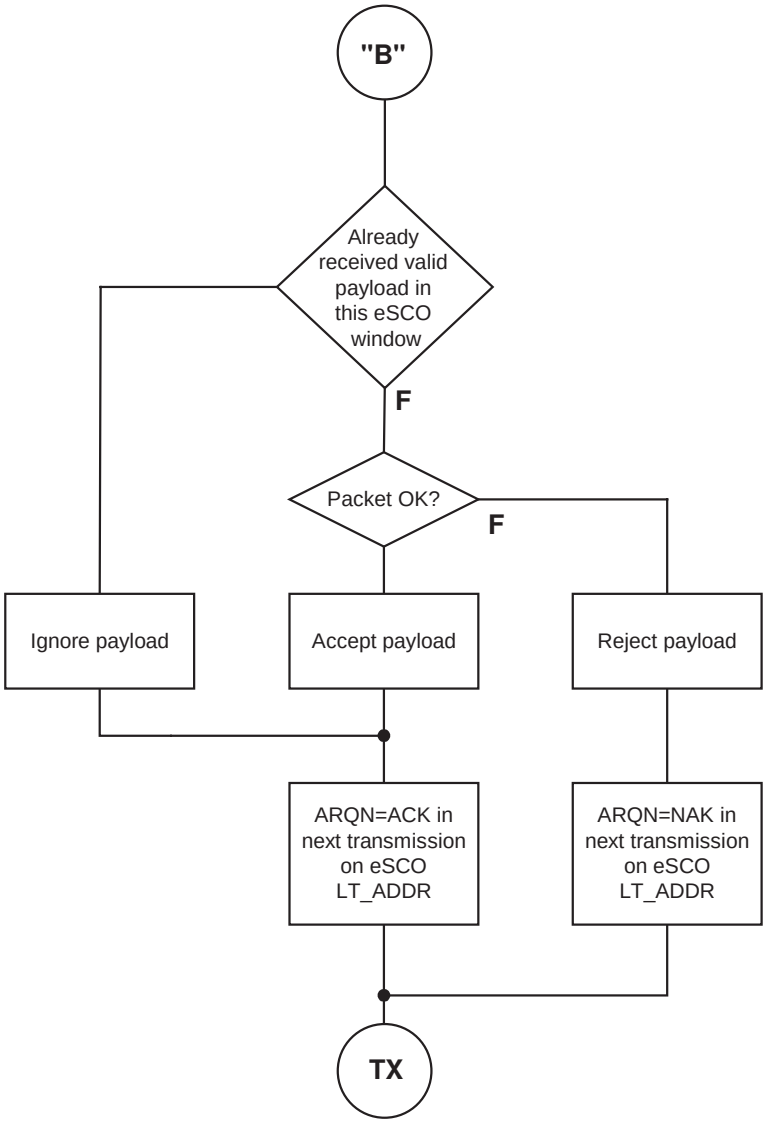


Figure 56—Stage 2 (eSCO) of the receive protocol for determining the ARQN bit

8.7.6.2 Retransmit filtering

The data payload shall be transmitted until a positive acknowledgment is received or a timeout is exceeded. A retransmission shall be carried out either because the packet transmission itself failed or because the acknowledgment transmitted in the return packet failed (note that the latter has a lower failure probability since the header is more heavily coded). In the latter case, the destination keeps receiving the same payload over and over again. In order to filter out the retransmissions in the destination, the SEQN bit is present in the header. Normally, this bit is alternated for every new CRC data payload transmission. In case of a retransmission, this bit shall not be changed so the destination can compare the SEQN bit with the previous SEQN value. If different, a new data payload has arrived; otherwise, it is the same data payload and may be ignored. Only new data payloads shall be transferred to the BB resource manager. CRC data payloads can be carried only by **DM**, **DH**, **DV**, or **EV** packets.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.7.6.2.1 Initialization of SEQN at start of new connection

The SEQN bit of the first CRC data packet at the start of a connection (as a result of page, page scan, role switch, or unpark) on both the master and the slave sides shall be set to 1. The subsequent packets shall use the rules in 8.7.6.2.2 through 8.7.6.2.5.

8.7.6.2.2 ACL and SCO retransmit filtering

The SEQN bit shall be affected only by the CRC data packets as shown in Figure 57. It shall be inverted every time a new CRC data packet is sent. The CRC data packet shall be retransmitted with the same SEQN number until an ACK is received or the packet is flushed. When an ACK is received, a new payload may be sent and on that transmission the SEQN bit shall be inverted. If a device decides to flush (see 8.7.6.3) and it has not received an acknowledgment for the current packet, it shall replace the current packet with an ACL-U continuation packet with the same SEQN as the current packet and length zero. If it replaces the current packet in this way, it shall not move on to transmit the next packet until it has received an ACK.

If the slave receives a packet other than **DH**, **DM**, **DV**, or **EV** with the SEQN bit inverted from that in the last header successfully received on the same LT_ADDR, it shall set the ARQN bit to NAK until a **DH**, **DM**, **DV** or **EV** packet is successfully received.

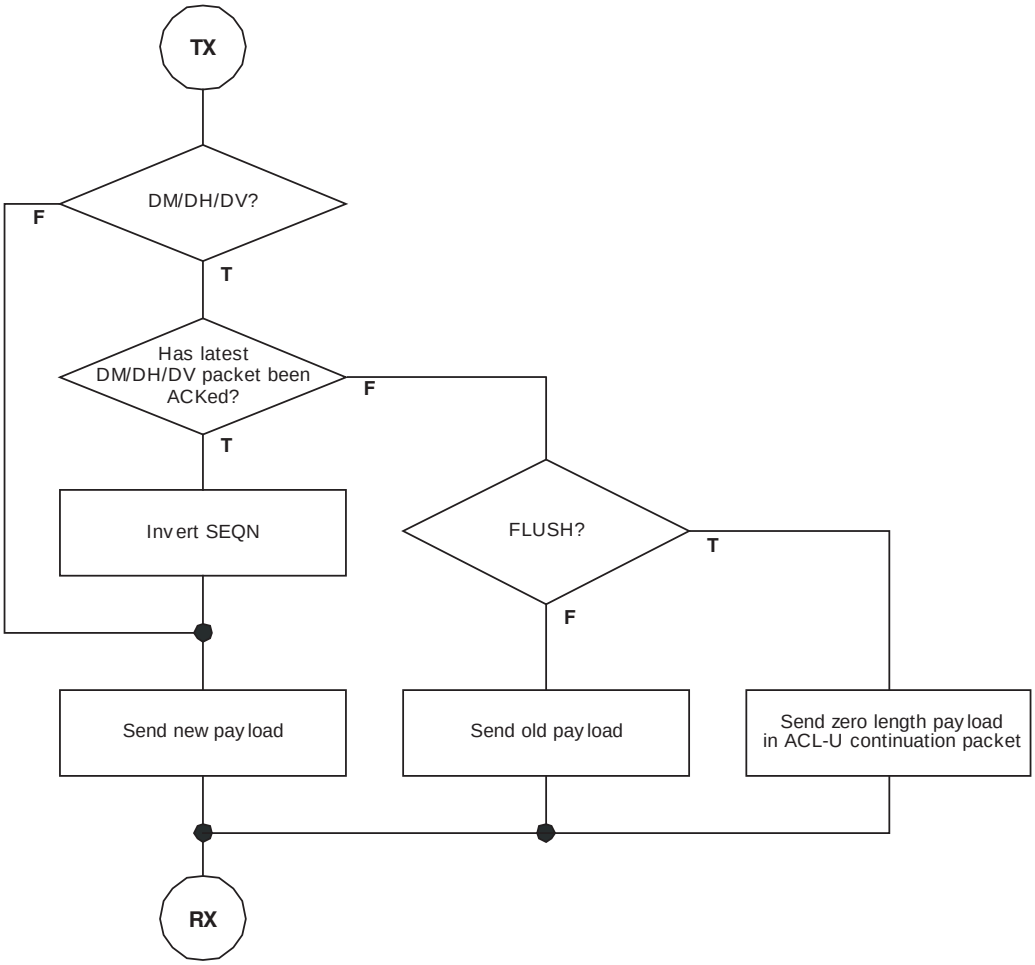


Figure 57—Transmit filtering for packets with CRC

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.7.6.2.3 eSCO retransmit filtering

In eSCO, the SEQN bit shall be toggled every eSCO window. The value shall be constant for the duration of the eSCO window. The initial value of SEQN shall be zero.

For a given eSCO window the SEQN value shall be constant.

8.7.6.2.4 FHS retransmit filtering

The SEQN bit in the FHS packet is not meaningful. This bit may be set to any value. Contents of the SEQN bit in the FHS packet shall not be checked.

8.7.6.2.5 Packets without CRC retransmit filtering

During transmission of packets without a CRC, the SEQN bit shall remain the same as it was in the previous packet.

8.7.6.3 Flushing payloads

In ACL, the ARQ scheme can cause variable delay in the traffic flow since retransmissions are inserted to assure error-free data transfer. For certain communication links, only a limited amount of delay is allowed: retransmissions are allowed up to a certain limit at which the current payload shall be ignored. This data transfer is indicated as *isochronous traffic*. This means that the retransmit process must be overruled in order to continue with the next data payload. Aborting the retransmit scheme is accomplished by flushing the old data and forcing the link controller to take the next data instead.

Flushing results in loss of remaining portions of an L2CAP message. Therefore, the packet following the flush shall have a start packet indication of LLID = 10 in the payload header for the next L2CAP message. This informs the destination of the flush. (see 8.6.6). Flushing will not necessarily result in a change in the SEQN bit value (see 8.7.6.2).

The flush timeout defines a maximum period after which all segments of the ACL-U packet are flushed from the controller buffer. The flush timeout shall start when the first segment of the ACL-U packet is stored in the controller buffer. After the flush timeout has expired, the link controller may continue transmissions according to the procedure described in 8.7.6.2.2; however, the BB resource manager shall not continue the transmission of the ACL-U packet to the link controller. If the BB resource manager has further segments of the packet queued for transmission to the link controller, it shall delete the remaining segments of the ACL-U packet from the queue. In case the complete ACL-U packet was not stored in the controller buffer yet, any continuation segments, received for the ACL logical transport, shall be flushed until the first segment of the next ACL-U packet for the logical transport is received. When the complete ACL-U packet has been flushed, the LM shall continue transmission of the next ACL-U packet for the ACL logical transport. The default flush timeout shall be infinite, i.e., retransmissions are carried out until physical link loss occurs. This is also referred to as a *reliable channel*. All devices shall support the default flush timeout.

In eSCO, packets shall be automatically flushed at the end of the eSCO window.

8.7.6.4 Multislave considerations

In a piconet with multiple logical transports, the master shall carry out the ARQ protocol independently on each logical transport.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.7.6.5 Broadcast packets

Broadcast packets are packets transmitted by the master to all the slaves simultaneously. If multiple hop sequences are being used, each transmission may be received only by some of the slaves. In this case the master shall repeat the transmission on each hop sequence. A broadcast packet shall be indicated by the all-zero LT_ADDR (note that the FHS packet is the only packet that may have an all-zero LT_ADDR, but is not a broadcast packet). Broadcast packets shall not be acknowledged.

Since broadcast messages are not acknowledged, each broadcast packet is transmitted at least a fixed number of times. A broadcast packet should be transmitted N_{BC} times before the next broadcast packet of the same broadcast message is transmitted (see Figure 58). Optionally, a broadcast packet may be transmitted $N_{BC} + 1$ times. Note that $N_{BC} = 1$ means that each broadcast packet should be sent only once, but optionally may be sent twice. However, time-critical broadcast information may abort the ongoing broadcast train. For instance, unpark messages sent at beacon instances may do this (see 8.8.9.5).

If multiple hop sequences are being used, then the master may transmit on the different hop sequences in any order, providing that transmission of a new broadcast packet shall not be started until all transmissions of any previous broadcast packet have completed on all hop sequences. The transmission of a single broadcast packet may be interleaved among the hop sequences to minimize the total time to broadcast a packet. The master has the option of transmitting only N_{BC} times on channels common to all hop sequences.

Broadcast packets with a CRC shall have their own SEQN. The SEQN of the first broadcast packet with a CRC shall be set to 1 by the master and shall be inverted for each new broadcast packet with CRC thereafter. Broadcast packets without a CRC have no influence on the SEQN. The slave shall accept the SEQN of the first broadcast packet it receives in a connection and shall check for change in SEQN for subsequent broadcast packets. Since there is no acknowledgment of broadcast messages and there is no end packet indication, it is important to receive the start packets correctly.[10] To ensure this, repetitions of the broadcast packets that are L2CAP start packets and LMP packets shall not be filtered out. These packets shall be indicated by LLID = 1X in the payload header as explained in 8.6.6. Only repetitions of the L2CAP continuation packets shall be filtered out.

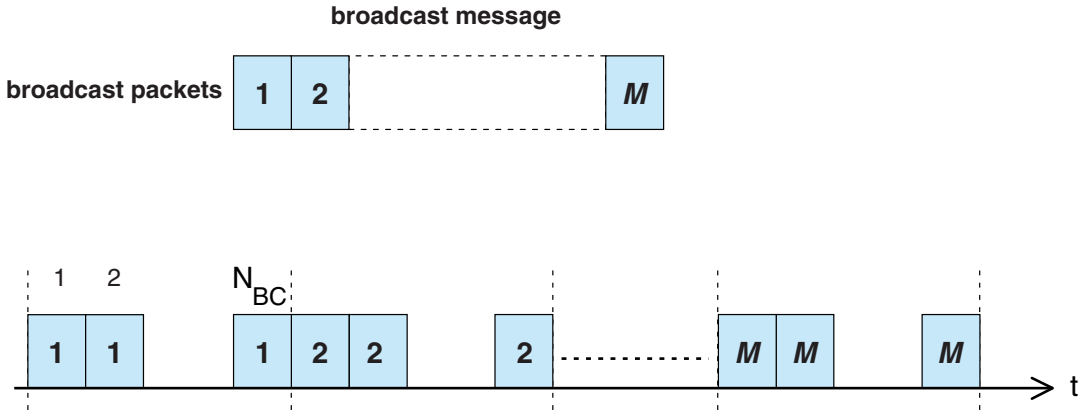


Figure 58—Broadcast repetition scheme

[10]Previous versions of the standard reset sequence numbering each time the first fragment of an ACL-U packet began broadcast. This meant that if an ACL-U packet fit into a single BB packet, it was impossible to distinguish it from the start of the next ACL-U packet. To avoid losing start fragments of ACL-U packets, the solution was to not filter out any start fragments of ACL-U packets. To keep behavior the same in all versions of the specification, this behavior has been retained.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8 Link controller operation

This subclause describes how a piconet is established and how devices can be added to and released from the piconet. Several states of operation of the devices are defined to support these functions. In addition, the operation of several piconets with one or more common members, the *scatternet*, is discussed.

8.8.1 Overview of states

Figure 59 shows a state diagram illustrating the different states used in the link controller. There are three major states: STANDBY, CONNECTION, and PARK; in addition, there are seven substates, **page**, **page scan**, **inquiry**, **inquiry scan**, **master response**, **slave response**, and **inquiry response**. The substates are interim states that are used to establish connections and enable device discovery. To move from one state or substate to another, either commands from the LM are used, or internal signals in the link controller are used (such as the trigger signal from the correlator and the timeout signals). It should be noted that Figure 62 reflects the fact that a device may have more than one simultaneous connection and may be in different states with respect to each connection. So, for example, a single connection cannot move straight from the **inquiry** substate to the CONNECTION state, but a device that has an active connection may hold or sniff that connection and then move to the **inquiry** substate.

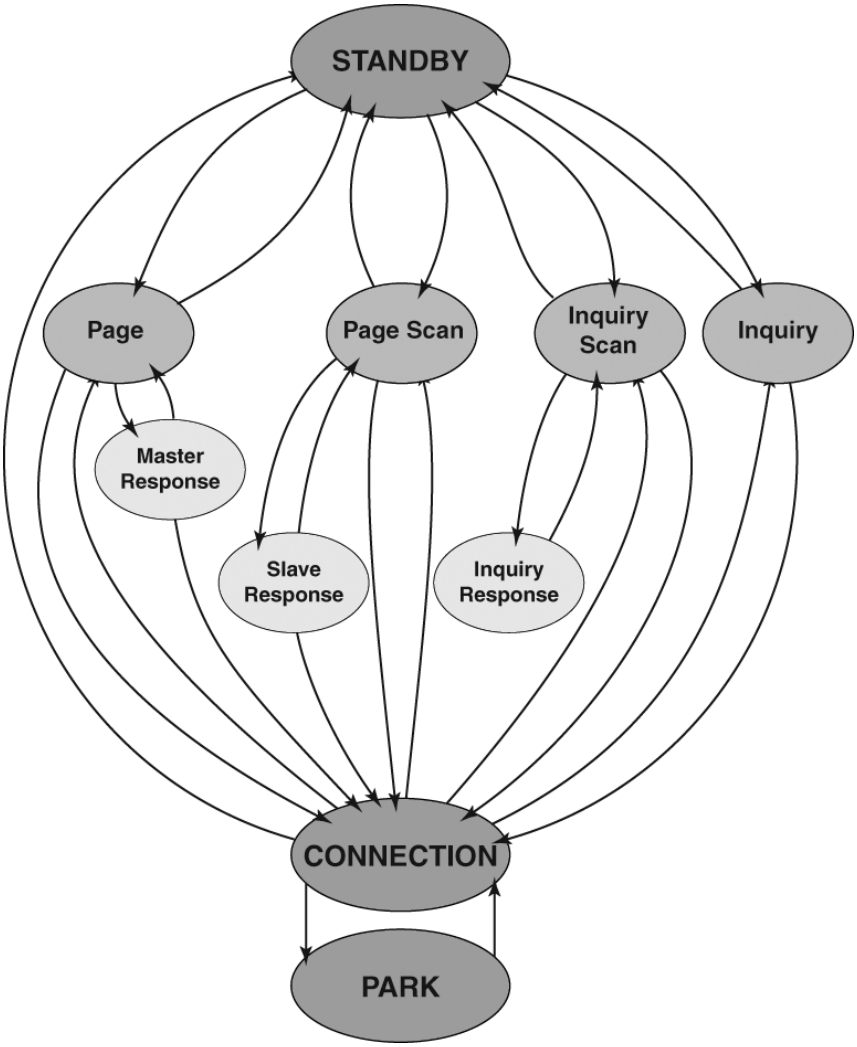


Figure 59—State diagram of link controller

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Figure 60 shows a typical progress through the states for a single connection.

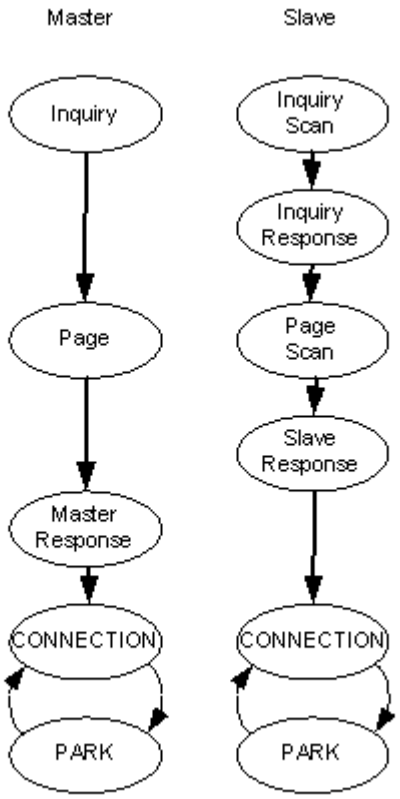


Figure 60—Typical progress though states

8.8.2 STANDBY state

The STANDBY state is the default state in the device. In this state, the device may be in a low-power mode. Only the CLKN is running at the accuracy of the LPO (or better).

The controller may leave the STANDBY state to scan for page or inquiry messages or to conduct a page or an inquiry.

8.8.3 Connection establishment substates

In order to establish new connections, the paging procedure is used. Only the device address is required to set up a connection. Knowledge about the clock, obtained from the inquiry procedure (see 8.8.4) or from a previous connection with this device, and the page scanning mode of the other device will accelerate the setup procedure. A device that establishes a connection carries out a page procedure and will automatically become the master of the connection.

8.8.3.1 Page scan substate

In the **page scan** substate, a device may be configured to use either the standard or interlaced scanning procedure. During a standard scan, a device listens for the duration of the scan window $T_{w\_page\_scan}$ (11.25 ms default) (see 11.7.3.20), while the interlaced scan is performed as two back-to-back scans of $T_{w\_page\_scan}$. If the scan interval is not at least twice the scan window, then interlaced scan shall not be used. During each scan window, the device shall listen at a single hop frequency, its correlator matched to its DAC. The scan window shall be long enough to completely scan 16 page frequencies.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When a device enters the **page scan** substate, it shall select the scan frequency according to the page hopping sequence determined by the device's device address (see 8.2.6.4.1). The phase in the sequence shall be determined by $CLKN_{16-12}$ of the device's CLKN, i.e., every 1.28 s a different frequency is selected.

In the case of a standard scan, if the correlator exceeds the trigger threshold during the page scan, the device shall enter the **slave response** substate described in 8.8.3.3.1. The scanning device may also use interlaced scan. In this case, if the correlator does not exceed the trigger threshold during the first scan, it shall scan a second time using the phase in the sequence determined by $[CLKN_{16-12} + 16] \bmod 32$. If, on this second scan, the correlator exceeds the trigger threshold, the device shall enter the **slave response** substate using $[CLKN_{16-12} + 16] \bmod 32$ as the frozen CLKN* in the calculation for $Xprs^{(79)}$ (see 8.2.6.4.3 for details). If the correlator does not exceed the trigger threshold during a scan in normal mode or during the second scan in interlaced scan mode, it shall return to either the STANDBY or CONNECTION state.

The **page scan** substate can be entered from the STANDBY state or the CONNECTION state. In the STANDBY state, no connection has been established, and the device can use all the capacity to carry out the page scan. Before entering the **page scan** substate from the CONNECTION state, the device should reserve as much capacity as possible for scanning. If desired, the device may place ACL connections in HOLD mode, PARK state, or SNIFF mode (see 8.8.8 and 8.8.9). Synchronous connections should not be interrupted by the page scan, although eSCO retransmissions should be paused during the scan. The page scan may be interrupted by the reserved synchronous slots, which should have higher priority than the page scan. SCO packets that require the least amount of capacity (**HV3** packets) should be used. The scan window shall be increased to minimize the setup delay. If one SCO logical transport is present using **HV3** packets and $T_{SCO} = 6$ slots or if one eSCO logical transport is present using **EV3** packets and $T_{ESCO} = 6$ slots, a total scan window $T_{w\_page\_scan}$ of at least 36 slots (22.5 ms) is recommended. If two SCO links are present using **HV3** packets and $T_{SCO} = 6$ slots or if two eSCO links are present using **EV3** packets and $T_{ESCO} = 6$ slots, a total scan window of at least 54 slots (33.75 ms) is recommended.

The scan interval $T_{page\ scan}$ is defined as the interval between the beginnings of two consecutive page scans. A distinction is made between the case where the scan interval is equal to the scan window $T_{w\_page\_scan}$ (continuous scan), the scan interval is maximal 1.28 s, or the scan interval is maximal 2.56 s. These three cases shall determine the behavior of the paging device, i.e., whether the paging device shall use R0, R1, or R2 (see also 8.8.3.2). Table 24 illustrates the relationship between $T_{page\ scan}$ and modes R0, R1, and R2. Although scanning in the R0 mode is continuous, the scanning may be interrupted, for example, by reserved synchronous slots. The scan interval information is included in the SR field in the FHS packet.

Table 24—Relationship between scan interval and paging modes R0, R1, and R2

SR mode	$T_{page\ scan}$
R0	≤ 1.28 s and = $T_{w\_page\_scan}$
R1	≤ 1.28 s
R2	≤ 2.56 s
Reserved	—

8.8.3.2 Page substate

The **page** substate is used by the master (source) to activate and connect to a slave (destination) in the **page scan** substate. The master tries to coincide with the slave's scan activity by repeatedly transmitting the paging message consisting of the slave's DAC in different hop channels. Since the clocks of the master and the slave are not synchronized, the master does not know exactly when the slave wakes up and on which hop

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

frequency. Therefore, it transmits a train of identical paging messages at different hop frequencies and listens between the transmit intervals until it receives a response from the slave.

The page procedure in the master consists of a number of steps. On systems with separate host and controller, the host first communicates the BD_ADDR of the slave to the controller. This BD_ADDR shall be used by the master to determine the page hopping sequence (see 8.2.6.4.2). The slave's BD_ADDR shall be used to determine the page hopping sequence (see 8.2.6.4.2). For the phase in the sequence, the master shall use an estimate of the slave's clock. For example, this estimate can be derived from timing information that was exchanged during the last encounter with this particular device (which could have acted as a master at that time) or from an inquiry procedure. With this CLKE of the slave's CLKN, the master can predict on which hop channel the slave starts page scanning.

The estimate of the clock in the slave can be completely wrong. Although the master and the slave use the same hopping sequence, they use different phases in the sequence and might never select the same frequency. To compensate for the clock drifts, the master shall send its page message during a short time interval on a number of wake-up frequencies. It shall transmit also on hop frequencies just before and after the current, predicted hop frequency. During each TX slot, the master shall sequentially transmit on two different hop frequencies. In the following RX slot, the receiver shall listen sequentially to two corresponding RX hops for an ID packet. The RX hops shall be selected according to the page response hopping sequence. The page response hopping sequence is strictly related to the page hopping sequence: for each page hop there is a corresponding page response hop. The RX/TX timing in the **page** substate is described in 8.2.2.5 (see also Figure 18). In the next TX slot, it shall transmit on two hop frequencies different from the former ones. The hop rate is increased to 3200 hop/s.

With the increased hopping rate as described above, the transmitter can cover 16 different hop frequencies in 16 slots or 10 ms. The page hopping sequence is divided over two paging trains, **A** and **B**, of 16 frequencies. Train **A** includes the 16 hop frequencies surrounding the current, predicted hop frequency $f(k)$, where k is determined by the clock estimate CLKE_{16-12}. The first train consists of hops

$$f(k-8), f(k-7), \ldots, f(k), \ldots, f(k+7)$$

When the difference between the clocks of the master and the slave is between -8×1.28 s and $+7 \times 1.28$ s, one of the frequencies used by the master will be the hop frequency to which the slave will listen. Since the master does not know when the slave will enter the **page scan** substate, the master has to repeat this train **A** N_{page} times or until a response is obtained, whichever is shorter. If the slave scan interval corresponds to R1, the repetition number is at least 128. If the slave scan interval corresponds to R2 or if the master has not previously read the slave's SR mode, the repetition number is at least 256. If the master has not previously read the slave's SR mode, it shall use $N_{\mathrm{page}} \geq 256$. Note that CLKE_{16-12} changes every 1.28 s; therefore, every 1.28 s, the trains will include different frequencies of the page hopping set.

When the difference between the clocks of the master and the slave is less than -8×1.28 s or larger than $+7 \times 1.28$ s, the remaining 16 hops are used to form the new 10 ms train **B.** The second train consists of hops

$$f(k-16), f(k-15), \ldots, f(k-9), f(k+8) \ldots, f(k+15)$$

Train **B** shall be repeated for N_{page} times. If no response is obtained, train **A** shall be tried again N_{page} times. Alternate use of train A and train B shall be continued until a response is received or the timeout *pageTO* is exceeded. If a response is returned by the slave, the master device enters the **master response** substate.

The **page** substate may be entered from the STANDBY state or the CONNECTION state. In the STANDBY state, no connection has been established, and the device can use all the capacity to carry out the page. Before entering the **page** substate from the CONNECTION state, the device should free as much capacity as possible for scanning. To ensure this, it is recommended that the ACL connections are put in HOLD mode or PARK state. However, the synchronous connections shall not be disturbed by the page. This means that the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

page will be interrupted by the reserved SCO and eSCO slots, which have higher priority than the page. In order to obtain as much capacity for paging, it is recommended to use the SCO packets that use the least amount of capacity (**HV3** packets). If SCO or eSCO links are present, the repetition number N_{page} of a single train shall be increased (see Table 25). Here it has been assumed that the **HV3** packets are used with an interval T_{SCO} = 6 slots or **EV3** packets are used with an interval of T_{ESCO} = 6 slots, which would correspond to a 64 kb/s synchronous link.

Table 25—Relationship between train repetition and paging modes R0, R1, and R2 when synchronous links are present

SR mode	No synchronous link	One synchronous link (HV3)	Two synchronous links (HV3)
R0	$N_{page} \geq 1$	$N_{page} \geq 2$	$N_{page} \geq 3$
R1	$N_{page} \geq 128$	$N_{page} \geq 256$	$N_{page} \geq 384$
R2	$N_{page} \geq 256$	$N_{page} \geq 512$	$N_{page} \geq 768$

The construction of the page train shall be independent of the presence of synchronous links. In other words, synchronous packets are sent on the reserved slots, but shall not affect the hop frequencies used in the unreserved slots (see Figure 61).

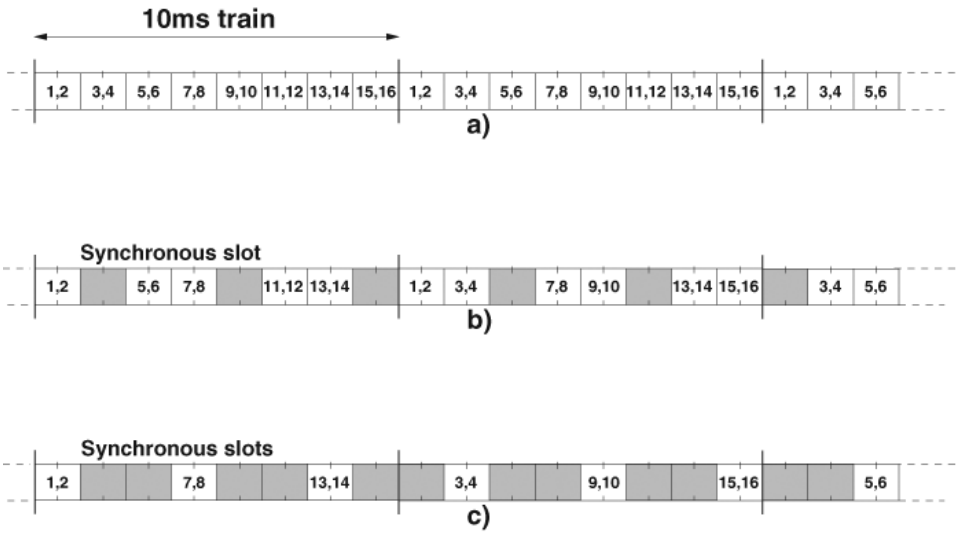


Figure 61—Conventional page (a), page while one synchronous link present (b), page while two synchronous links present (c)

8.8.3.3 Page response substates

When a page message is successfully received by the slave, there is a coarse frequency hopping synchronization between the master and the slave. Both the master and the slave enter a response substate to exchange vital information to continue the connection setup. It is important for the piconet connection that both devices shall use the same CAC, that they shall use the same channel hopping sequence, and that their clocks are synchronized. These parameters shall be derived from the master device. The device that initializes the connection (starts paging) is defined as the master device (which is thus valid only during the time the piconet exists). The CAC and channel hopping sequence shall be derived from the device address (BD_ADDR)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

of the master. The timing shall be determined by CLK. An offset shall be added to the slave's CLNE to temporarily synchronize the slave clock to the CLK. At startup, the master parameters are transmitted from the master to the slave. The messaging between the master and the slave at startup is specified in this subclause.

The initial messaging between master and slave is shown in Table 26 and in Figure 62 and Figure 63. In those two figures, frequencies $f(k)$, $f(k + 1)$, etc., are the frequencies of the page hopping sequence determined by the slave's BD_ADDR. The frequencies $f'(k)$, $f'(k + 1)$, etc., are the corresponding page response frequencies (slave-to-master). The frequencies $g(m)$ belong to the basic channel hopping sequence.

Table 26—Initial messaging during startup

Step	Message	Packet type	Direction	Hopping sequence	Access code and clock
1	Page	ID	Master to slave	Page	Slave
2	First slave page response	ID	Slave to master	Page response	Slave
3	Master page response	FHS	Master to slave	Page	Slave
4	Second slave page response	ID	Slave to master	Page response	Slave
5	1st packet master	POLL	Master to slave	Channel	Master
6	1st packet slave	Any type	Slave to master	Channel	Master

In step 1 (see Table 26), the master device is in **page** substate, and the slave device in the **page scan** substate. Assume in this step that the page message sent by the master reaches the slave. On receiving the page message, the slave enters the **slave response** substate in step 2. The master waits for a reply from the slave, and when this arrives in step 2, it will enter the **master response** substate in step 3. Note that during the initial message exchange, all parameters are derived from the slave's device address and that only the page hopping and page response hopping sequences are used (these are also derived from the slave's device address). Note that when the master and slave enter the response substates, their clock input to the page and page response hop selection is frozen as is described in 8.2.6.4.3.

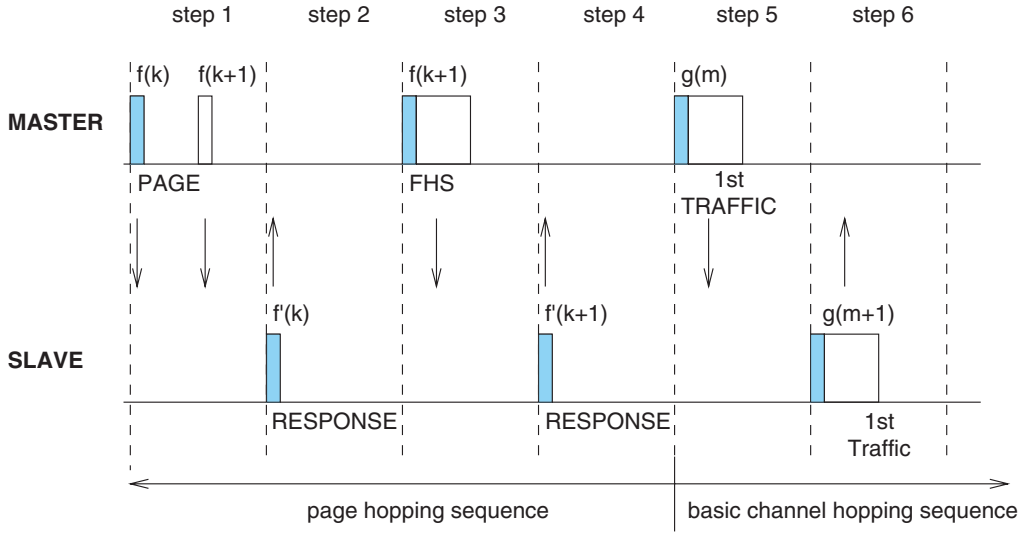


Figure 62—Messaging at initial connection when slave responds to first page message

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

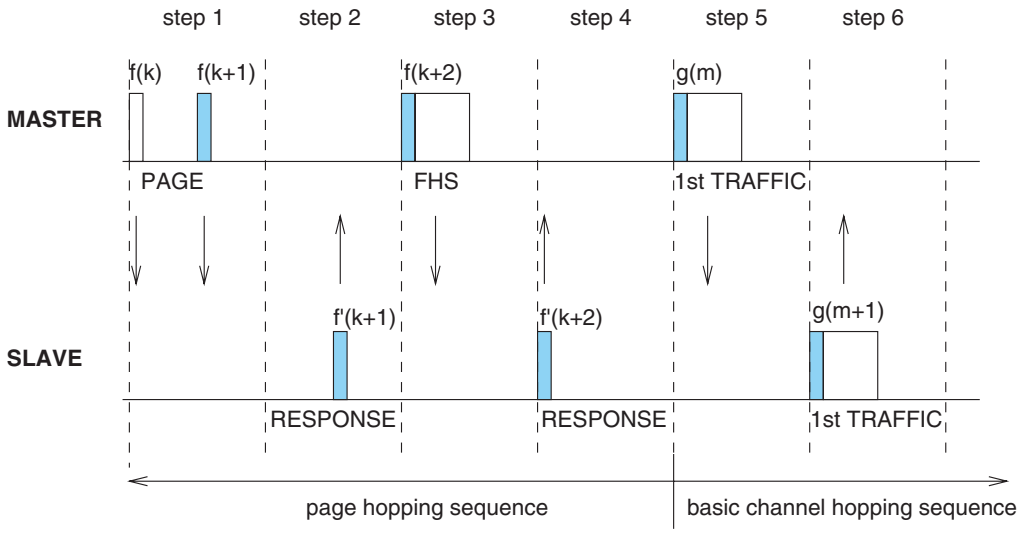


Figure 63—Messaging at initial connection when slave responds to second page message

8.8.3.3.1 Slave response substate

After having received the page message in step 1, the slave device shall transmit a slave page response message (the slave's DAC) in step 2. This response message shall be the slave's DAC. The slave shall transmit this response 625 μs after the beginning of the received page message and at the response hop frequency that corresponds to the hop frequency in which the page message was received. The slave transmission is, therefore, time aligned to the master transmission. During initial messaging, the slave shall still use the page response hopping sequence to return information to the master. The clock input $CLKN_{16-12}$ shall be frozen at the value it had at the time the page message was received.

After having sent the response message, the slave's receiver shall be activated 312.5 μs after the start of the response message and shall await the arrival of an FHS packet. Note that an FHS packet can arrive 312.5 μs after the arrival of the page message as shown in Figure 63 and not after 625 μs as is usually the case in the piconet physical channel RX/TX timing. More details about the timing can be found in 8.2.4.4.

If the setup fails before the CONNECTION state has been reached, the following procedure shall be carried out. The slave shall listen as long as no FHS packet is received until *pagerespTO* is exceeded. Every 1.25 ms, however, it shall select the next master-to-slave hop frequency according to the page hop sequence. If nothing is received after *pagerespTO*, the slave shall return back to the **page scan** substate for one scan period. Length of the scan period depends on the synchronous reserved slots present. If no page message is received during this additional scan period, the slave shall resume scanning at its regular scan interval and return to the state it was in prior to the first page scan state.

If an FHS packet is received by the slave in the **slave response** substate, the slave shall return a slave page response message in step 4 to acknowledge reception of the FHS packet. This response shall use the page response hopping sequence. The transmission of the slave page response packet is based on the reception of the FHS packet. Then the slave shall change to the master's CAC and clock as received from the FHS packet. Only the 26 MSBs of the CLK are transferred: the timing shall be such that CLK_1 and CLK_0 are both zero at the time the FHS packet was received as the master transmits in even slots only. The offset between the master's clock and the slave's clock shall be determined from the master's clock in the FHS packet and reported to the slave's BB resource manager.

Finally, the slave enters the CONNECTION state in step 5. From then on, the slave shall use the master's clock and the master's BD_ADDR to determine the basic channel hopping sequence and the CAC. The slave

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

shall use the LT_ADDR in the FHS payload as the primary LT_ADDR in the CONNECTION state. The connection mode shall start with a POLL packet transmitted by the master. The slave may respond with any type of packet. If the POLL packet is not received by the slave or the response packet is not received by the master within *newconnectionTO* number of slots after FHS packet acknowledgment, the master and the slave shall return to **page** and **page scan** substates, respectively. See 8.8.5.

8.8.3.3.2 Master response substate

When the master has received a slave page response message in step 2, it shall enter the master response routine. It shall freeze the current clock input to the page hop selection scheme. The master shall then transmit an FHS packet in step 3 containing the master's CLKN, the master's BD_ADDR, the BCH parity bits, and the class of device. The FHS packet contains all information to construct the CAC without requiring a mathematical derivation from the master's device address. The LT_ADDR field in the packet header of FHS packets in the **master response** substate shall be set to all zeros. The FHS packet shall be transmitted at the beginning of the master-to-slave slot following the slot in which the slave responded. The FHS packet shall carry the all-zero LT_ADDR. The TX timing of the FHS packet is not based on the reception of the response packet from the slave. The FHS packet may, therefore, be sent 312.5 μs after the reception of the response packet as shown in Figure 63 and not 625 μs after the received packet as is usual in the piconet physical channel RX/TX timing (see also 8.2.4.4).

After the master has sent its FHS packet, it shall wait for a second slave page response message in step 4 acknowledging the reception of the FHS packet. This response shall be the slave's DAC. If no response is received, the master shall retransmit the FHS packet with an updated clock and still using the slave's parameters. It shall retransmit the FHS packet with the clock updated each time until a second slave page response message is received or the timeout of *pageRespTO* is exceeded. In the latter case, the master shall return to the **page** substate and send an error message to the BB resource manager. During the retransmissions of the FHS packet, the master shall use the page hopping sequence.

If the slave's response is received, the master shall change to using the master parameters, so it shall use the CAC and the CLK. The FHS packet transmission shall start when the lower clock bits CLK0 and CLK1 are reset to zero. These clock bits are not included in the FHS packet. Finally, the master enters the CONNECTION state in step 5. The master BD_ADDR shall be used to change to a new hopping sequence, the *basic channel hopping sequence*. The basic channel hopping sequence uses all 79 hop channels in a pseudo-random fashion (see also 8.2.6.4.7). The master shall now send its first traffic packet in a hop determined with the new (master) parameters. This first packet shall be a POLL packet. See 8.8.5. This packet shall be sent within *newconnectionTO* number of slots after reception of the FHS packet acknowledgment. The slave may respond with any type of packet. If the POLL packet is not received by the slave or the POLL packet response is not received by the master within *newconnectionTO* number of slots, the master and the slave shall return to **page** and **page scan** substates, respectively.

8.8.4 Device discovery substates

In order to discover other devices, a device shall enter the **inquiry** substate. In this substate, it shall repeatedly transmit the inquiry message (ID packet) (see 8.6.5.1.1) at different hop frequencies. The inquiry hop sequence is derived from the LAP of the GIAC. Thus, even when DIACs are used, the applied hopping sequence is generated from the GIAC LAP. A device that allows itself to be discovered should regularly enter the **inquiry scan** substate to respond to inquiry messages. The message exchange and contention resolution during inquiry response are described in 8.8.4.1 through 8.8.4.3. The inquiry response is optional: a device is not forced to respond to an inquiry message.

During the **inquiry** substate, the discovering device collects the device addresses and clocks of all devices that respond to the inquiry message. It may then, if desired, make a connection to any one of them by means of the page procedure described in 8.8.3.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The inquiry message broadcast by the source does not contain any information about the source. However, it may indicate which classes of device should respond. There is one GIAC to inquire for any device and a number of DIACs that inquire only for a certain type of device. The IACs are derived from reserved device addresses and are further described in 8.6.3.1.

8.8.4.1 Inquiry scan substate

The **inquiry scan** substate is very similar to the **page scan** substate. However, instead of scanning for the device's DAC, the receiver shall scan for the IAC long enough to completely scan for 16 inquiry frequencies. Two types of scans are defined: standard and interlaced. In the case of a standard scan, the length of this scan period is denoted $T_{w\_inquiry\_scan}$ (11.25 ms default) (see 11.7.3.22). The standard scan is performed at a single hop frequency as defined by Xir_{4-0} (see 8.2.6.4.6). The interlaced scan is performed as two back-to-back scans of $T_{w\_inquiry\_scan}$ where the first scan is on the normal hop frequency and the second scan is defined by $[Xir_{4-0} + 16] \bmod 32$. If the scan interval is not at least twice the scan window, then interlaced scan shall not be used. The inquiry procedure uses 32 dedicated inquiry hop frequencies according to the inquiry hopping sequence. These frequencies are determined by the general inquiry address. The phase is determined by CLKN of the device carrying out the inquiry scan; the phase changes every 1.28 s.

Instead of, or in addition to, the GIAC, the device may scan for one or more DIACs. However, the scanning shall follow the inquiry scan hopping sequence determined by the general inquiry address. If an inquiry message is received during an inquiry wake-up period, the device shall enter the **inquiry response** substate.

The **inquiry scan** substate can be entered from the STANDBY state or the CONNECTION state. In the STANDBY state, no connection has been established, and the device can use all the capacity to carry out the inquiry scan. Before entering the **inquiry scan** substate from the CONNECTION state, the device should reserve as much capacity as possible for scanning. If desired, the device may place ACL logical transports in SNIFF mode, HOLD mode, or PARK state. Synchronous logical transports are preferably not interrupted by the inquiry scan, although eSCO retransmissions should be paused during the scan. In this case, the inquiry scan may be interrupted by the reserved synchronous slots. SCO packets that require the least amount of capacity (**HV3** packets) should be used. The scan window $T_{w\_inquiry\_scan}$ shall be increased to increase the probability to respond to an inquiry message. If one SCO logical transport is present using **HV3** packets and $T_{SCO} = 6$ slots or if one eSCO logical transport is present using **EV3** packets and $T_{ESCO} = 6$ slots, a total scan window of at least 36 slots (22.5 ms) is recommended. If two SCO links are present using **HV3** packets and $T_{SCO} = 6$ slots or if two eSCO links are present using **EV3** packets and $T_{ESCO} = 6$ slots, a total scan window of at least 54 slots (33.75 ms) is recommended.

The scan interval $T_{inquiry\_scan}$ is defined as the interval between two consecutive inquiry scans. The inquiry scan interval shall be less than or equal to 2.56 s.

8.8.4.2 Inquiry substate

The **inquiry** substate is used to discover new devices. This substate is very similar to the **page** substate; the TX/RX timing shall be the same as in paging (see 8.2.4.4 and Figure 18). The TX and RX frequencies shall follow the inquiry hopping sequence and the inquiry response hopping sequence and are determined by the GIAC and the CLKN of the discovering device. Between inquiry transmissions, the receiver shall scan for inquiry response messages. When a response is received, the entire packet (an FHS packet) is read, after which the device shall continue with inquiry transmissions. The device in an **inquiry** substate shall not acknowledge the inquiry response messages. If enabled by the host (see 11.7.3.54), the RSSI value of the inquiry response message shall be measured. The device shall keep probing at different hop channels and between, listening for response packets. As in the **page** substate, two 10 ms trains, **A** and **B**, are defined, splitting the 32 frequencies of the inquiry hopping sequence into two 16-hop parts. A single train shall be repeated for at least $N_{inquiry} = 256$ times before a new train is used, except for the first train in a series, which may be repeated fewer times. In order to collect all responses in an error-free environment, at least three train switches must have taken place. As a result, the **inquiry** substate may have to last for 10.24 s unless the

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

inquirer collects enough responses and aborts the **inquiry** substate earlier. If desired, the inquirer may also prolong the **inquiry** substate to increase the probability of receiving all responses in an error-prone environment. If an inquiry procedure is automatically initiated periodically (e.g., a 10 s period every minute), then the interval between two inquiry instances shall be determined randomly. This is done to avoid two devices synchronizing their inquiry procedures.

The **inquiry** substate is continued until stopped by the BB resource manager (when it decides that it has sufficient number of responses), when a timeout has been reached (*inquiryTO*), or by a command from the host to cancel the inquiry procedure.

The **inquiry** substate can be entered from the STANDBY state or the CONNECTION state. In the STANDBY state, no connection has been established, and the device can use all the capacity to carry out the inquiry. Before entering the **inquiry** substate from the CONNECTION state, the device should free as much capacity as possible for scanning. To ensure this, it is recommended that the ACL logical transports are placed in SNIFF mode, HOLD mode, or PARK state. However, the reserved slots of synchronous logical transports shall not be disturbed by the inquiry. This means that the inquiry will be interrupted by the reserved SCO and eSCO slots, which have higher priority than the inquiry. In order to obtain as much capacity as possible for inquiry, it is recommended to use the SCO packets that use the least amount of capacity (**HV3** packets). If SCO or eSCO links are present, the repetition number N_{inquiry} shall be increased (see Table 27). Here it has been assumed that **HV3** packets are used with an interval $T_{\text{SCO}} = 6$ slots or **EV3** packets are used with an interval of $T_{\text{ESCO}} = 6$ slots, which would correspond to a 64 kb/s synchronous link.

Table 27—Increase of train repetition when synchronous links are present

	No synchronous links	One synchronous link (HV3)	Two synchronous links (HV3)
N_{inquiry}	≥ 256	≥ 512	≥ 768

8.8.4.3 Inquiry response substate

The **slave response** substate for inquiries differs completely from the **slave response** substate applied for pages. When the inquiry message is received in the **inquiry scan** substate, the recipient shall return an inquiry response (FHS) packet containing the recipient's device address (BD_ADDR) and other parameters.

The following protocol in the slave's inquiry response shall be used. On the first inquiry message received in this substate, the slave shall enter the **inquiry response** substate and shall return an FHS response packet to the master 625 μs after the inquiry message was received. A contention problem may arise when several devices are in close proximity to the inquiring device and all respond to an inquiry message at the same time. However, because every device has a free-running clock, it is highly unlikely that they all use the same phase of the inquiry hopping sequence. In order to avoid repeated collisions between devices that wake up in the same inquiry hop channel simultaneously, a device shall back off for a random period of time. Thus, if the device receives an inquiry message and returns an FHS packet, it shall generate a random number, RAND, between 0 and MAX_RAND. For scanning intervals ≥ 1.28 s, MAX_RAND shall be 1023; however, for scanning intervals < 1.28 s, MAX_RAND may be as small as 127. A profile that uses a special DIAC may choose to use a smaller MAX_RAND than 1023 even when the scanning interval is ≥ 1.28 s. The slave shall return to the CONNECTION or STANDBY state for the duration of at least RAND time slots. Before returning to the CONNECTION and STANDBY state, the device may go through the **page scan** substate. After at least RAND slots, the device shall add an offset of 1 to the phase in the inquiry hop sequence (the phase has a 1.28 s resolution) and return to the **inquiry scan** substate again. If the slave is triggered again, it shall repeat the procedure using a new random number. The offset to the clock accumulates each time an FHS packet is returned. During a probing window, a slave may respond multiple times, but on

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

different frequencies and at different times. Reserved synchronous slots should have priority over response packets. In other words, if a response packet overlaps with a reserved synchronous slot, it shall not be sent, but the next inquiry message is awaited.

The messaging during the inquiry routines is summarized in Table 28. In step 1, the master transmits an inquiry message using the IAC and its own clock. The slave responds with the FHS packet containing the slave's device address, CLKN, and other slave information. This FHS packet is returned at times that tend to be random. The FHS packet is not acknowledged in the inquiry routine, but it is retransmitted at other times and frequencies as long as the master is probing with inquiry messages.

Table 28—Messaging during inquiry routines

Step	Message	Packet type	Direction	Hopping sequence	Access code and clock
1	Inquiry	ID	Master to slave	Inquiry	Inquiry
2	Inquiry response	FHS	Slave to master	Inquiry response	Inquiry

8.8.5 Connection state

In the CONNECTION state, the connection has been established, and packets can be sent back and forth. In both devices, the CAC (master), the CLK, and the AFH_channel_map are used. The CONNECTION state uses the basic or adapted channel hopping sequence.

The CONNECTION state starts with a POLL packet sent by the master to verify the switch to the master's timing and channel frequency hopping. The slave may respond with any type of packet. If the slave does not receive the POLL packet or the master does not receive the response packet for *newconnectionTO* number of slots, both devices shall return to **page**/**page scan** substates.

The first information packets in the CONNECTION state contain control messages that characterize the link and give more details regarding the devices. These messages are exchanged between the LMs of the devices. For example, they may define the SCO logical transport and the sniff parameters. Then, the transfer of user information can start by alternately transmitting and receiving packets.

The CONNECTION state is left through a detach or reset command. The detach command is used if the link has been disconnected in the normal way; all configuration data in the link controller shall remain valid. The reset command is a soft reset of the link controller. The functionality of the soft reset is described in 11.7.3.2.

In the CONNECTION state, if a device is not going to be nominally present on the channel at all times, it may describe its unavailability by using SNIFF mode or HOLD mode (see Figure 64).

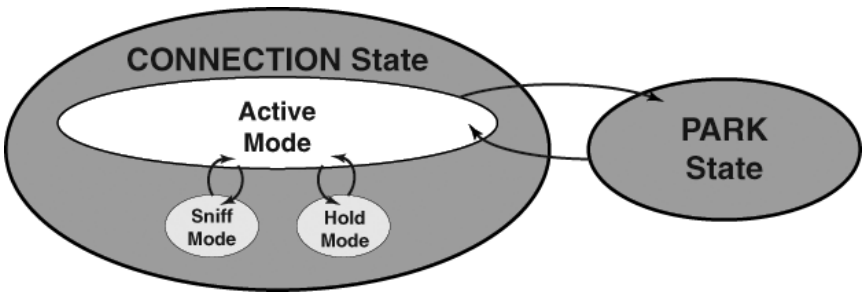


Figure 64—Connection state

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8.6 Active mode

In the active mode, both master and slave actively participate on the channel. Up to seven slaves may be in the active mode at any given time. The master schedules the transmissions based on traffic demands to and from the different slaves. In addition, it supports regular transmissions to keep slaves synchronized to the channel. Slaves in the active mode listen in the master-to-slave slots for packets. These devices are known as *active slaves*. If an active slave is not addressed, it may sleep until the next new master transmission. Slaves can derive the number of slots the master has reserved for transmission from the TYPE field in the packet header; during this time, the nonaddressed slaves do not have to listen on the master-to-slave slots. When a device is participating in multiple piconets, it should listen in the master-to-slave slot for the current piconet. It is recommended that a device not be away from each piconet in which it is participating for more than T_{poll} slots. A periodic master transmission is required to keep the slaves synchronized to the channel. Since the slaves need only the CAC to synchronize, any packet type can be used for this purpose.

Only the slave that is addressed by one of its LT_ADDRs (primary or secondary) may return a packet in the next slave-to-master slot. If no valid packet header is received, the slave may respond only in its reserved SCO or eSCO slave-to-master slot. In the case of a broadcast message, no slave shall return a packet (an exception is the access window for access requests in the PARK state; see 8.8.9).

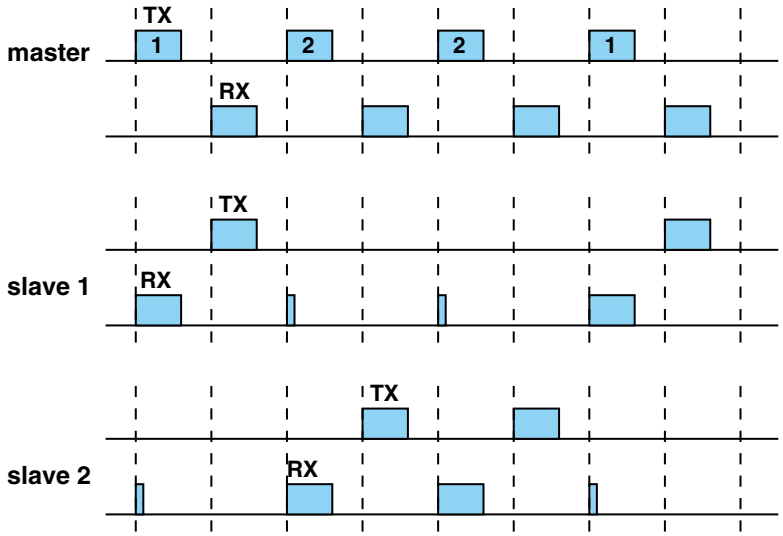


Figure 65—RX/TX timing in multislave configuration

8.8.6.1 Polling in the active mode

The master always has full control over the piconet. Due to the TDD scheme, slaves can communicate only with the master and not with other slaves. In order to avoid collisions on the ACL logical transport, a slave is allowed to transmit in the slave-to-master slot only when addressed by the LT_ADDR in the packet header in the preceding master-to-slave slot. If the LT_ADDR in the preceding slot does not match or a valid packet header was not received, the slave shall not transmit unless the slot is reserved for the slave on a synchronous logical transport.

The master normally attempts to poll a slave's ACL logical transport no less often than once every T_{poll} slots. T_{poll} is set by the LM (see 9.3.1.8).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The slave's ACL logical transport may be polled with any packet type except for FHS and ID. For example, polling during SCO may use **HV** packets, since the slave may respond to an **HV** packet with a **DM1** packet (see 8.8.6.2).

8.8.6.2 SCO

The SCO logical transport shall be established by the master sending an SCO setup message via the LMP. This message contains timing parameters including the SCO interval T_{SCO} and the offset D_{SCO} to specify the reserved slots.

In order to prevent clock wraparound problems, an initialization flag in the LMP setup message indicates whether initialization procedure 1 or 2 is being used. The slave shall apply the initialization method as indicated by the initialization flag. The master shall use initialization 1 when the MSB of the current master clock CLK_{27} is 0; it shall use initialization 2 when the MSB of the current master clock CLK_{27} is 1. The master-to-slave SCO slots reserved by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

$CLK_{27-1} \bmod T_{SCO} = D_{SCO}$ for initialization 1

$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{SCO} = D_{SCO}$ for initialization 2

The slave-to-master SCO slots shall directly follow the reserved master-to-slave SCO slots. After initialization, the clock value $CLK(k + 1)$ for the next master-to-slave SCO slot shall be derived by adding the fixed interval T_{SCO} to the clock value of the current master-to-slave SCO slot:

$$CLK(k + 1) = CLK(k) + T_{SCO}$$

The master will send SCO packets to the slave at regular intervals (the SCO interval T_{SCO} counted in slots) in the reserved master-to-slave slots. An **HV1** packet can carry 1.25 ms of speech at a 64 kb/s rate. An **HV1** packet shall, therefore, be sent every two time slots ($T_{SCO} = 2$). An **HV2** packet can carry 2.5 ms of speech at a 64 kb/s rate. An **HV2** packet shall, therefore, be sent every four time slots ($T_{SCO} = 4$). An **HV3** packet can carries 3.75 ms of speech at a 64 kb/s rate. An **HV3** packet shall, therefore, be sent every six time slots ($T_{SCO} = 6$).

The slave is allowed to transmit in the slot reserved for its SCO logical transport unless the (valid) LT_ADDR in the preceding slot indicates a different slave. If no valid LT_ADDR can be derived in the preceding slot, the slave may still transmit in the reserved SCO slot.

Since the **DM1** packet is recognized on the SCO logical transport, it may be sent during the SCO reserved slots if a valid packet header with the primary LT_ADDR is received in the preceeding slot. **DM1** packets sent during SCO reserved slots shall be used only to send ACL-C data.

The slave shall not transmit anything other than an **HV** packet in a reserved SCO slot unless it decodes its own slave address in the packet header of the packet in the preceding master-to-slave transmission slot.

8.8.6.3 eSCO

The eSCO logical transport is established by the master sending an eSCO setup message via the LMP. This message contains timing parameters including the eSCO interval T_{ESCO} and the offset D_{ESCO} to specify the reserved slots.

To enter eSCO, the master or slave shall send an eSCO setup command via the LMP. This message shall contain the eSCO interval T_{ESCO} and an offset D_{ESCO}. In order to prevent clock wraparound problems, an

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

initialization flag in the LMP setup message indicates whether initialization procedure 1 or 2 shall be used. The slave shall apply the initialization method as indicated by the initialization flag. The master shall use initialization 1 when the MSB of the current master clock CLK_{27} is 0; it shall use initialization 2 when the MSB of the current master clock CLK_{27} is 1. The master-to-slave eSCO slots reserved by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

$CLK_{27-1} \bmod T_{ESCO} = D_{ESCO}$ for initialization 1

$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{ESCO} = D_{ESCO}$ for initialization 2

The slave-to-master eSCO slots shall directly follow the reserved master-to-slave eSCO slots. After initialization, the clock value $CLK(k + 1)$ for the next master-to-slave eSCO slot shall be found by adding the fixed interval T_{ESCO} to the clock value of the current master-to-slave eSCO slot:

$$CLK(k + 1) = CLK(k) + T_{ESCO}$$

When an eSCO logical transport is established, the master shall assign an additional LT_ADDR to the slave. This provides the eSCO logical transport with an ARQ scheme that is separate from that of the ACL logical transport. All traffic on a particular eSCO logical transport, and only that eSCO traffic, is carried on the eSCO LT_ADDR. The eSCO ARQ scheme uses the ARQN bit in the packet header and operates similarly to the ARQ scheme on ACL links.

There are two different polling rules in eSCO. In the eSCO reserved slots, the polling rule is the same as the rule used in the SCO reserved slots. The master may send a packet in the master slot. The slave may transmit on the eSCO LT_ADDR in the following slot either if it received a packet on the eSCO LT_ADDR in the previous slot or if it did not receive a valid packet header in the previous slot. When the master-to-slave packet type is a three-slot packet, the slave's transmit slot is the fourth slot of the eSCO reserved slots. A master shall transmit in an eSCO retransmission window on a given eSCO LT_ADDR only if it addressed that eSCO LT_ADDR in the immediately preceeding eSCO reserved slots. A slave may transmit on an eSCO LT_ADDR in the eSCO reserved slots only if the slave did not received a valid packet header with a different LT_ADDR in the eSCO reserved slots. Inside retransmission windows, the same polling rule as for ACL traffic is used: the slave shall transmit on the eSCO channel only if it received a valid packet header and correct LT_ADDR on the eSCO channel in the previous master-to-slave transmission slot. The master may transmit on any non-eSCO LT_ADDR in any master-to-slave transmission slot inside the eSCO retransmission window. The master shall transmit on an eSCO LT_ADDR in the retransmission window only if there are enough slots left for both the master and slave packets to complete in the retransmission window. The master may refrain from transmitting in any slot during the eSCO retransmission window. When there are no data to transmit from master to slave, either due to the traffic being unidirectional or due to the master-to-slave packet having been acknowledged, the master shall use the POLL packet. When the master-to-slave packet has been acknowledged and the slave-to-master packet has been correctly received, the master shall not address the slave on the eSCO LT_ADDR until the next eSCO reserved slot, with the exception that the master may transmit a NULL packet with ARQN = ACK on the eSCO LT_ADDR. When there are no data to transmit from slave to master, either due to the traffic being unidirectional or due to the slave-to-master packet having been acknowledged, the slave shall use NULL packets. eSCO traffic should be given priority over ACL traffic in the retransmission window.

Figure 66 shows the eSCO window when single-slot packets are used.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

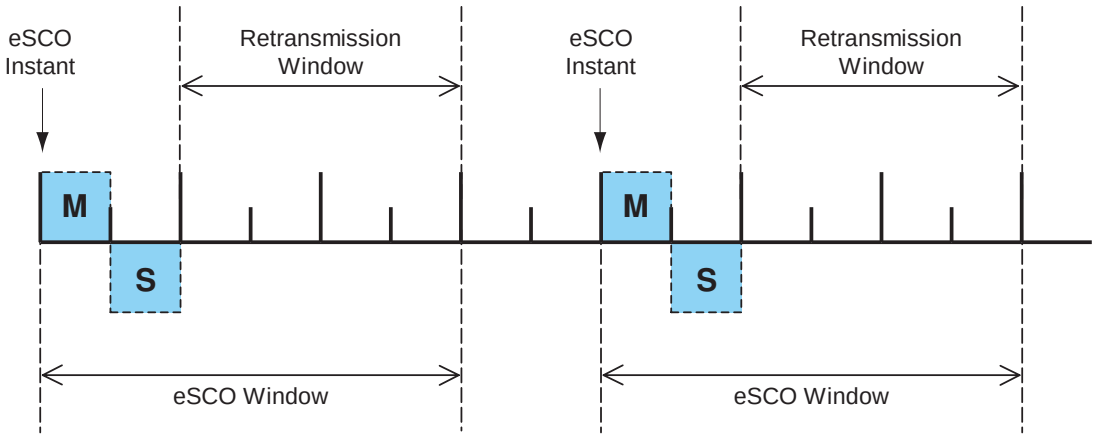


Figure 66—eSCO window details for single-slot packets

When multislot packets are used in either direction of the eSCO logical transport, the first transmission continues into the following slots. The retransmission window in this case starts the slot after the end of the slave-to-master packet, i.e., two, four, or six slots immediately following the eSCO instant are reserved and should not be used for other traffic. Figure 67 shows the eSCO window when multislot packets are used in one direction and single-slot packets are used in the other direction.

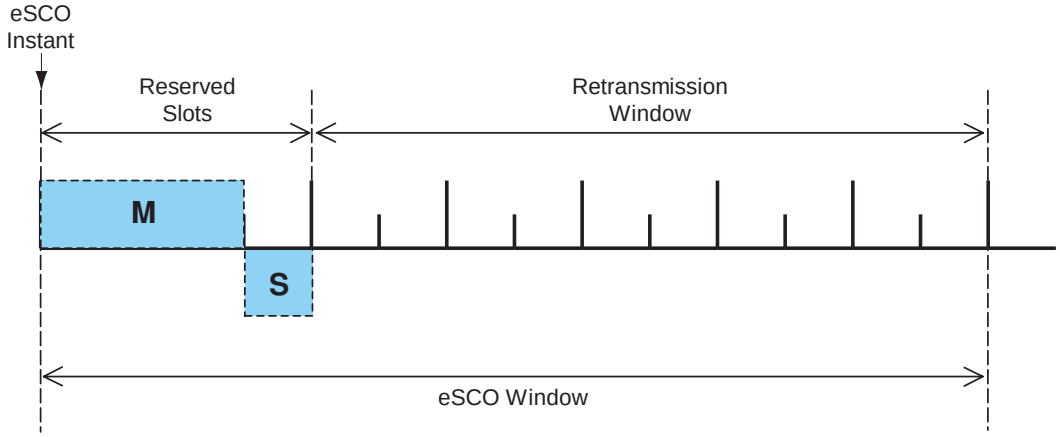


Figure 67—eSCO window details for asymmetric traffic

eSCO windows may overlap on the master, but shall not overlap on an individual slave.

In the reserved slots, both master and slave should listen and transmit at their allocated slots at the first transmission time of each eSCO window. Intermittent lapses due to, for instance, time-critical signaling during connection establishment are allowed. Both master and slave may refrain from sending data and may use instead POLL and NULL packets, respectively. When the master transmits a POLL packet instead of an **EV4** or **EV5** packet, the slave shall transmit, starting in the same slot as if the master transmitted an **EV4** or **EV5** packet. If the slave does not receive anything in the reserved master-to-slave transmission slot, it shall transmit in the same slot as if the master had transmitted the negotiated packet type. For example, if the master had negotiated an **EV5** packet, the slave would transmit three slots later. If the master does not receive a slave transmission in response to an eSCO packet, it causes an implicit NAK of the packet in question. The listening requirements for the slave during the retransmission window are the same as for an active ACL logical transport.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8.6.4 Broadcast scheme

The master of the piconet can broadcast messages to all slaves. A broadcast packet shall have an LT_ADDR set to all zero. Each new broadcast message (which may be carried by a number of packets) shall start with the start of L2CAP message indication (LLID = 10).

A broadcast packet shall never be acknowledged. In an error-prone environment, the master may carry out a number of retransmissions to increase the probability for error-free delivery (see also 8.7.6.5).

In order to support the PARK state (as described in 8.8.9), a master transmission shall take place at fixed intervals. This master transmission will act as a beacon to which slaves can synchronize. If no traffic takes place at the beacon event, broadcast packets shall be sent. More information is given in 8.8.9.

8.8.6.5 Role switch

There are several occasions when a role switch is used:

— A role switch is necessary when joining an existing piconet by paging since, by definition, the paging device is initially master of a “small” piconet involving only the pager (master) and the paged (slave) device.
— A role switch is necessary in order for a slave in an existing piconet to set up a new piconet with itself as master and the original piconet master as slave. If the original piconet had more than one slave, then this implies a double role for the original piconet master; it becomes a slave in the new piconet while still maintaining the original piconet as master.

Prior to the role switch, encryption, if present, shall be stopped in the old piconet. A role switch shall not be performed if the physical link is in SNIFF mode, HOLD mode, or PARK state or has any synchronous logical transports.

For the master and slave involved in the role switch, the switch results in a reversal of their TX and RX timing: a *TDD switch*. Additionally, since the piconet parameters are derived from the device address and clock of the master, a role switch inherently involves a redefinition of the piconet as well: a *piconet switch*. The new piconet’s parameters shall be derived from the former slave’s device address and clock.

Assume device A is to become master; device B was the former master. Then there are two alternatives: either the slave initiates the role switch or the master initiates the role switch. These alternatives are described in 9.3.4.2. The BB procedure is the same regardless of which alternative is used.

In step 1, the device A (slave) and devide B (master) shall perform a TDD switch using the former hopping scheme (still using the device address and clock of device B) so there is no piconet switch yet. The slot offset information sent by device A shall not be used yet, but shall be used in step 3. Device A now becomes the master; device B, the slave. The LT_ADDR formerly used by device A in its slave role shall now be used by devide B (now slave).

At the moment of the TDD switch, both devices A and B shall start a timer with a timeout of *newconnectionTO*. The timer shall be stopped in device B (slave) as soon as it receives an FHS packet from device A (master) on the TDD-switched channel. The timer shall be stopped in device A (master) as soon as it receives an ID packet from device B (slave). If the *newconnectionTO* expires, the master and slave shall return to the old piconet timing and AFH state, taking their old roles of master and slave. The FHS packet shall be sent by device A (master) using the old piconet parameters. The LT_ADDR in the FHS packet header shall be the former LT_ADDR used by device A. The LT_ADDR carried in the FHS payload shall be the new LT_ADDR intended for device B when operating on the new piconet. After the FHS acknowledgment, which is the ID packet and shall be sent by device B (slave) on the old hopping sequence, both device A (master) and device B (slave) shall use the new channel parameters of the new piconet as indicated by the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

FHS with the sequence selection set to basic channel hopping sequence. If the new master has physical links that are AFH enabled, following the piconet switch, the new master is responsible for controlling the AFH operational mode of its new slave.

Since the old and new masters' clocks are synchronized, the clock information sent in the FHS payload shall indicate the new master's clock at the begining of the FHS packet transmission. Furthermore, the 1.25 ms resolution of the clock information given in the FHS packet is not sufficient for aligning the slot boundaries of the two piconets. The slot offset information in the LMP message previously sent by device A shall be used to provide more accurate timing information. The slot offset indicates the delay between the start of the master-to-slave slots of the old and new piconet physical channels. This timing information ranges from 0 to 1249 μs with a resolution of 1 μs. It shall be used together with the clock information in the FHS packet to accurately position the correlation window when switching to the new master's timing after acknowledgment of the FHS packet.

After reception of the FHS packet acknowledgment, device A (new master) shall switch to its own timing with the sequence selection set to the basic channel hopping sequence and shall send a POLL packet to verify the switch. Both the master and the slave shall start a new timer with a timeout of *newconnectionTO* on FHS packet acknowledgment. The start of this timer shall be aligned with the beginning of the first master TX slot boundary of the new piconet, following the FHS packet acknowledgment. The slave shall stop the timer when the POLL packet is received; the master shall stop the timer when the POLL packet is acknowledged. The slave shall respond with any type of packet to acknowledge the POLL. Any pending AFH_Instant shall be cancelled once the POLL packet has been received by the slave. If no response is received, the master shall resend the POLL packet until *newconnectionTO* is reached. If this timer expires, both the slave and the master shall return to the old piconet timing with the old master and slave roles. Expiry of the timer shall also restore the state associated with AFH (including any pending AFH_Instant), CQDDR (see 9.3.1.7), and power control (see 9.3.1.3). The procedure may then start again beginning at step 1. Aligning the timer with TX boundaries of the new piconet ensures that no device returns to the old piconet timing in the middle of a master RX slot.

After the role switch, the ACL logical transport is reinitialized as if it were a new connection. For example, the SEQN of the first data packet containing a CRC on the new piconet physical channel shall be set according to the rules in 8.7.6.2.

A parked slave must be unparked before it can participate in a role switch.

8.8.6.6 Scatternet

Multiple piconets may cover the same area. Since each piconet has a different master, the piconets hop independently, each with their own hopping sequence and phase as determined by the respective master. In addition, the packets carried on the channels are preceded by different CACs as determined by the master device addresses. As more piconets are added, the probability of collisions increases; a graceful degradation of performance results as is common in frequency hopping spread spectrum (FHSS) systems.

If multiple piconets cover the same area, a device can participate in two or more overlaying piconets by applying time multiplexing. To participate on the proper channel, it shall use the associated master device address and proper clock offset to obtain the correct phase. A device can act as a slave in several piconets, but as a master in only a single piconet: since two piconets with the same master are synchronized and use the same hopping sequence, they are one and the same piconet. A group of piconets in which connections exist between different piconets is called a *scatternet*.

A master or slave can become a slave in another piconet by being paged by the master of the other piconet. On the other hand, a device participating in one piconet can page the master or slave of another piconet. Since the paging device always starts out as master, a master-slave role switch is required if a slave role is desired. This is described in the 8.8.6.5.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8.6.6.1 Interpiconet communications

Time multiplexing must be used to switch between piconets. Devices may achieve the time multiplexing necessary to implement scatternet by using SNIFF mode or by remaining in an active ACL connection. For an ACL connection in piconets where the device is a slave in the CONNECTION state, the device may choose to not listen in every master slot. In this case, it should be recognized that the QoS on this link may degrade abruptly if the slave is not present enough to match up with the master polling that slave. Similarly, in piconets where the device is master, the device may choose to not transmit in every master slot. In this case, it is important to honor T_{poll} as much as possible. Devices in SNIFF mode may have sufficient time to visit another piconet between sniff slots. When the device is a slave using SNIFF mode and there are not sufficient idle slots, the device may choose to not listen to all master transmission slots in the sniff_attempts period or during the subsequent sniff_timeout period. A master is not required to transmit during sniff slots and, therefore, has flexibility for scatternet. If SCO or eSCO links are established, other piconets shall be visited only in the nonreserved slots between reserved slots. This is possible only if there is a single SCO logical transport using **HV3** packets or eSCO links where at least four slots remain between the reserved slots. Since the multiple piconets are not synchronized, guard time must be left to account for misalignment. This means that only two slots can effectively be used to visit another piconet between the **HV3** packets. Since clocks are unsycnhronized and accurate only to ± 20 ppm when a device is active, or ± 250 ppm in low-power modes, the unsynchronized clocks of piconets will drift with respect to one another. This means that if a device has a synchronous link in one piconet, the timing of slots available to it in another piconet will drift. If SNIFF mode is used to manage bandwidth in the piconets, then sniff instants may have to be periodically renegotiated.

Since the clocks of two masters of different piconets are not synchronized, a slave device participating in two piconets shall maintain two offsets that, added to its own CLKN, create the two CLKs. Since the two CLKs drift independently, the slave must regularly update the offsets in order to keep synchronization to both masters.

8.8.6.7 Hop sequence switching

Hop sequence adaptation is controlled by the master and can be set to either enabled or disabled. Once enabled, hop sequence adaptation shall apply to all logical transports on a physical link. Once enabled, the master may periodically update the set of used and unused channels as well as disable hop sequence adaptation on a physical link. When a master has multiple physical links, the state of each link is independent of all other physical links.

When hop sequence adaptation is enabled, the sequence selection hop selection kernel input is set to adapted channel hopping sequence and the input AFH_channel_map is set to the appropriate set of used and unused channels. Additionally, the same channel mechanism shall be used. When hop sequence adaptation is enabled with all channels used, this is known as AHS(79).

When hop sequence adaptation is disabled, the sequence selection input of the hop selection kernel is set to basic channel hopping sequence (the input AFH_channel_map is unused in this case), and the same channel mechanism shall not be used.

The hop sequence adaptation state shall be changed when the master sends the LMP_set_AFH PDU and a BB acknowledgment is received. When the BB acknowledgment is received prior to the hop sequence switch instant AFH_Instant (see 9.3.1.4), the hop sequence proceeds as shown in Figure 68.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

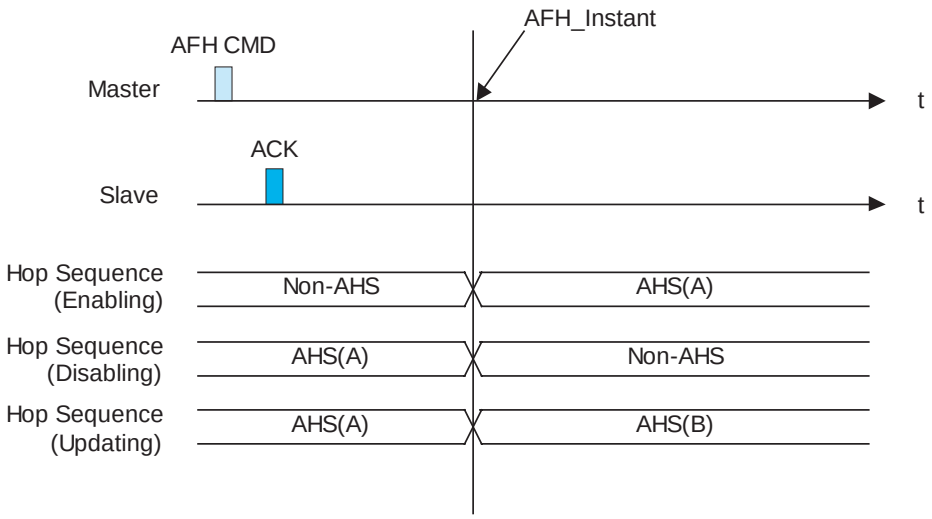


Figure 68—Successful hop sequence switching

When the BB acknowledgment is not received prior to the AFH_Instant, the master shall use a recovery hop sequence for the slave(s) that did not respond with an acknowledgment (this may be because the slave did not hear the master's transmission or the master did not hear the slave's transmission). When hop sequence adaptation is being enabled or disabled, the recovery sequence shall be the AFH_channel_map specified in the LMP_set_AFH PDU. When the AFH_channel_map is being updated, the master shall choose a recovery sequence that includes all of the RF channels marked as used in either the old or new AFH_channel_map, e.g., AHS(79). Once the BB acknowledgment is received, the master shall use the AFH_channel_map in the LMP_set_AFH PDU starting with the next transmission to the slave. See Figure 69.

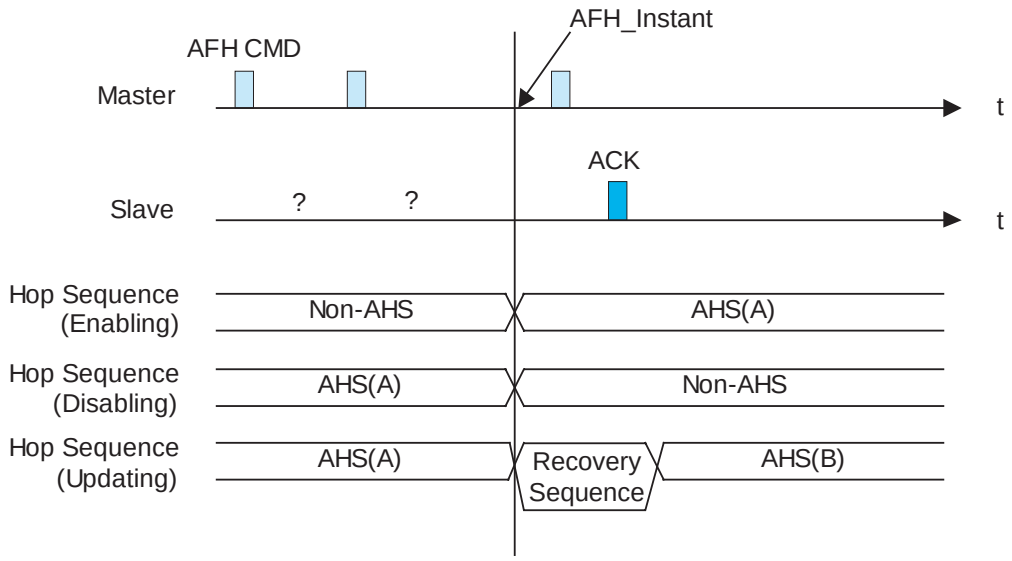


Figure 69—Recovery hop sequences

When the AFH_Instant occurs during a multislot packet transmitted by the master, the slave shall use the same hopping sequence parameters as the master used at the start of the multislot packet. An example of this is shown in Figure 70. In this figure the basic channel hopping sequence is designated *f*. The first adapted channel hopping sequence is designated with *f′*, and the second adapted channel hopping sequence is designated *f″*.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

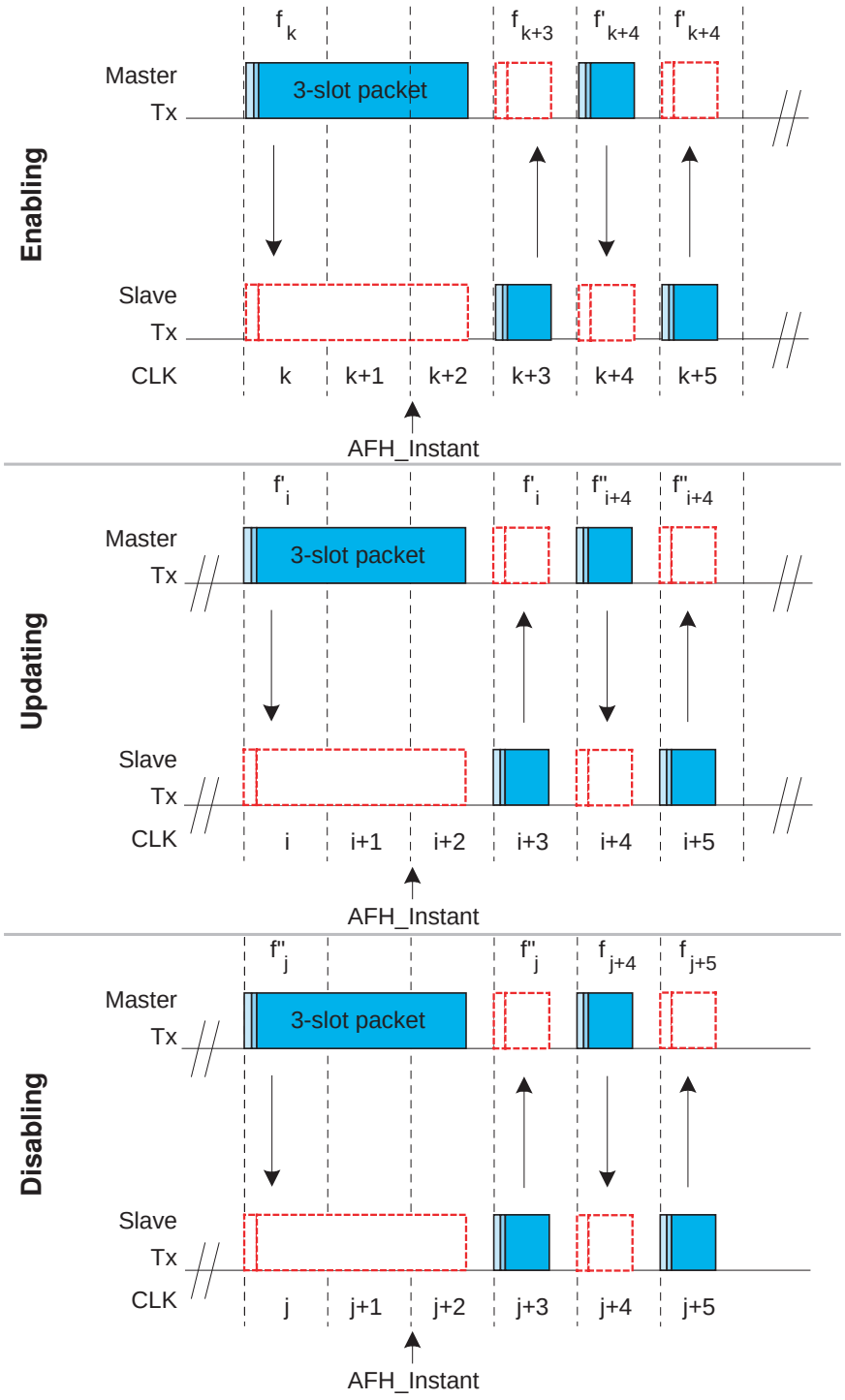


Figure 70—AFH_Instant changes during multislot packets transmitted by the master

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8.6.8 Channel classification and channel map selection

RF channels are classified as being unknown, bad, or good. These classifications are determined individually by the master and slaves based on local information (e.g., active or passive channel assessment methods or from the host via HCI). Information received from other devices via LMP (e.g., an AFH_channel_map from a master or a channel classification report from a slave) shall not be included in a device's channel classification.

The three possible channel classifications shall be as defined in Table 29.

Table 29—Channel classification descriptions

Classification	Definition
Unknown	The channel assessment measurements are insufficient to reliably classify the channel, and the channel is not marked as bad in the most recent HCI Set_AFH_Channel_Classification command.
Bad	a) An ACL or synchronous throughput failure measure associated with a channel has exceeded a threshold (defined by the particular channel assessment algorithm employed). b) An interference-level measure associated with a channel, determining the interference level that the link poses upon other systems in the vicinity, has exceeded a threshold (defined by the particular channel assessment algorithm employed). c) A channel is marked as bad in the most recent HCI Set_AFH_Channel_Classification command.
Good	A channel is neither unknown or bad.

A master with AFH-enabled physical links shall determine an AFH_channel_map based on any combination of the following information:

— Channel classification from local measurements (e.g., active or passive channel assessment in the controller), if supported and enabled. The host may enable or disable local measurements using the HCI Write_AFH_Channel_Classification_Mode command, defined in 11.7.3.58, if HCI is present.
— Channel classification information from the host using the HCI Set_AFH_Channel_Classification command, defined in 11.7.3.58, if HCI is present. Channels classified as bad in the most recent AFH_Host_Channel_Classification command shall be marked as unused in the AFH_channel_map.
— Channel classification reports received from slaves in LMP_channel_classification PDUs, defined in 9.3.1.5.

The algorithm used by the master to combine these information sources and generate the AFH_channel_map is not defined in this standard and will be implementation specific. At no time shall the number of channels used be less than N_{min}, defined in 8.2.3.1.

If a master determines that all channels should be used, it may keep AFH operation enabled using an AFH_channel_map of 79 used channels, i.e., AHS(79).

8.8.6.9 Power management

Features are provided to allow low-power operation. These features are both at the microscopic level when handling the packets and at the macroscopic level when using certain operation modes.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8.6.9.1 Packet handling

In order to minimize power consumption, packet handling is minimized both at TX and RX sides. At the TX side, power is minimized by sending only useful data. This means that if only link control information needs to be exchanged, NULL packets may be used. No transmission is required if there is no link control information to be sent or if the transmission would involve only a NAK (NAK is implicit on no reply). If there are data to be sent, the payload length shall be adapted in order to send only the valid data bytes. At the RX side, packet processing takes place in different steps. If no valid access code is found in the search window, the transceiver may return to sleep. If an access code is found, the receiver device shall start to process the packet header. If the HEC fails, the device may return to sleep after the packet header. A valid header indicates if a payload will follow and how many slots are involved.

8.8.6.9.2 Slot occupancy

As was described in 8.6.5, the packet type indicates how many slots a packet may occupy. A slave not addressed in the packet header may go to sleep for the remaining slots the packet occupies. This can be read from the TYPE code.

8.8.6.9.3 Recommendations for low-power operation

The most common and flexible methods for reducing power consumption are the use of SNIFF mode and PARK state. HOLD mode can also be used by repeated negotiation of HOLD periods.

8.8.7 SNIFF mode

In SNIFF mode, the duty cycle of the slave's activity in the piconet may be reduced. If a slave is in active mode on an ACL logical transport, it shall listen in every even ACL slot to that master traffic, unless

— A previous transmission indicated the slot would be occupied by a multislot packet for another slave, or
— That link is being treated as a scatternet link, or
— The device is absent due to HOLD mode

With SNIFF mode, the time slots when a slave is listening are reduced, so the master shall transmit to a slave only in specified time slots. The sniff anchor points are spaced regularly with an interval of $T_{\text{sniff.}}$

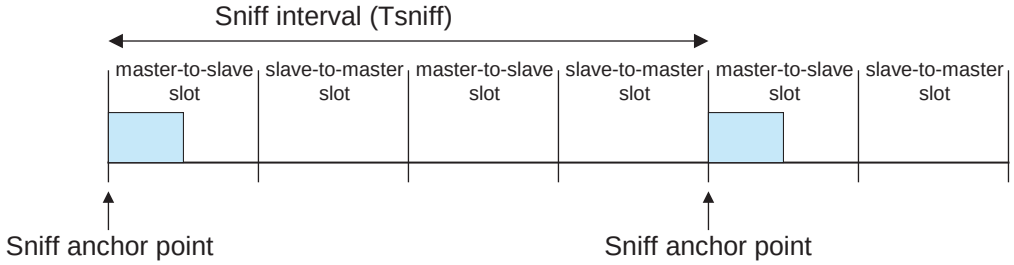


Figure 71—Sniff anchor points

The slave listens in master-to-slave transmission slots starting at the sniff anchor point. It shall continue listening if any of the following events occurs:

— If fewer than $N_{\text{sniff\_attempt}}$ master-to-slave transmission slots have elapsed since the sniff anchor point.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— If the slave has received a packet with a matching LT_ADDR that contains ACL data (**DM**, **DH**, **DV**, or **AUX1** packets) in the preceding $N_{sniff\_timeout}$ master-to-slave transmission slots.
— If the slave has transmitted a packet containing ACL data (**DM**, **DH**, **DV**, or **AUX1** packets) in the preceding $N_{sniff\_timeout}$ slave-to-master transmission slots.

A slave may continue listening if the slave has received any packet with a matching LT_ADDR in the preceding $N_{sniff\_timeout}$ master-to-slave transmission slots. Also, a device may override the rules above and stop listening prior to $N_{sniff\_timeout}$ or the remaining $N_{sniff\_attempt}$ slots if it has activity in another piconet.

It is possible that activity from one sniff timeout may extend to the next sniff anchor point. Any activity from a previous sniff timeout shall not affect activity after the next sniff anchor point. So in the above rules, only the slots since the last sniff anchor point are considered.

Note that $N_{sniff\_attempt} = 1$ and $N_{sniff\_timeout} = 0$ cause the slave to listen only at the slot beginning at the sniff anchor point, irrespective of packets received from the master.

$N_{sniff\_attempt} = 0$ shall not be used.

SNIFF mode applies only to asynchronous logical transports and their associated LT_ADDR. SNIFF mode shall not apply to synchronous logical transports; therefore, both masters and slaves shall still respect the reserved slots and retransmission windows of synchronous links.

To enter SNIFF mode, the master or slave shall issue a sniff command via the LMP. This message includes the sniff interval T_{sniff} and an offset D_{sniff}. In addition, an initialization flag indicates whether initialization procedure 1 or 2 shall be used. The device shall use initialization 1 when the MSB of the current master clock CLK_{27} is 0; it shall use initialization 2 when the MSB of the current master clock CLK_{27} is 1. The slave shall apply the initialization method as indicated by the initialization flag irrespective of its clock bit value CLK_{27}. The sniff anchor point determined by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

$CLK_{27-1} \bmod T_{sniff} = D_{sniff}$ for initialization 1

$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{sniff} = D_{sniff}$ for initialization 2

This implies that D_{sniff} must be even.

After initialization, the clock value CLK(k + 1) for the next sniff anchor point shall be derived by adding the fixed interval T_{sniff} to the clock value of the current sniff anchor point:

$CLK(k + 1) = CLK(k) + T_{sniff}$

8.8.7.1 SNIFF TRANSITION mode

SNIFF TRANSITION mode is a special mode that is used during the transition between SNIFF and active modes. It is required because during this transition the mode (SNIFF or active) in which the slave is located is unclear and it is necessary to ensure that the slave is polled correctly regardless of the mode in which it is located.

In SNIFF TRANSITION mode, the master shall maintain the active mode poll interval in case the slave is in active mode. In addition the master shall poll the slave at least once in the sniff attempt transmit slots starting at each sniff instant. Note that this transmission counts for the active mode polling as well. The master must use its high-power accurate clock when in SNIFF TRANSITION mode.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The precise circumstances under which the master enters SNIFF TRANSITION mode are defined in 9.3.5.3.1.

8.8.8 HOLD mode

During the CONNECTION state, the ACL logical transport to a slave can be put in a HOLD mode. In HOLD mode, the slave temporarily shall not support ACL packets on the channel. Any synchronous packet during reserved synchonous slots (from SCO and eSCO links) shall be supported. With the HOLD mode, capacity can be made free to do other things like scanning, paging, inquiring, or attending another piconet. The device in HOLD mode can also enter a low-power sleep mode. During HOLD mode, the slave device keeps its LT_ADDR(s).

Prior to entering HOLD mode, master and slave agree on the time duration the slave remains in HOLD mode. A timer shall be initialized with the *holdTO* value. When the timer is expired, the slave shall wake up, synchronize to the traffic on the channel, and wait for further master transmissions.

8.8.9 PARK state

When a slave does not need to participate on the piconet physical channel, but still needs to remain synchronized to the channel, it can enter PARK state. PARK state is a state with very little activity in the slave. In the PARK state, the slave shall give up its LT_ADDR. Instead, it shall receive two new addresses to be used in the PARK state:

— PM_ADDR: 8-bit parked member address
— AR_ADDR: 8-bit access request address

The PM_ADDR distinguishes a parked slave from the other parked slaves. This address may be used in the master-initiated unpark procedure. In addition to the PM_ADDR, a parked slave may also be unparked by its 48-bit BD_ADDR. The all-zero PM_ADDR is a reserved address: if a parked device has the all-zero PM_ADDR, it can be unparked only by the BD_ADDR. In that case, the PM_ADDR has no meaning. The AR_ADDR shall be used by the slave in the slave-initiated unpark procedure. All messages sent to the parked slaves are carried by broadcast packets.

The parked slave wakes up at regular intervals to listen to the channel in order to resynchronize and to check for broadcast messages. To support the synchronization and channel access of the parked slaves, the master supports a beacon train described in 8.8.9.1. The beacon structure is communicated to the slave when it is parked. When the beacon structure changes, the parked slaves are updated through broadcast messages.

The master shall maintain separate, nonoverlapping park beacon structures for each hop sequence. The beacon structures shall not overlap either their beacon slots or their access windows.

In addition to using PARK state for low power consumption, PARK state is used to connect more than seven slaves to a single master. At any one time, only seven slaves can be in the CONNECTION state. However, by swapping active and parked slaves in and out, respectively, of the piconet, the number of slaves can be much larger (255 if the PM_ADDR is used, and an arbitrarily large number if the BD_ADDR is used).

8.8.9.1 Beacon train

To support parked slaves, the master establishes a beacon train when one or more slaves are parked. The beacon train consists of one beacon slot or a train of equidistant beacon slots that is transmitted periodically with a constant time interval. The beacon train is illustrated in Figure 72. A train of N_B ($N_B \geq 1$) beacon slots is defined with an interval of T_B slots. The beacon slots in the train are separated by Δ_B. The start of the first beacon slot is referred to as the *beacon instant* and serves as the beacon timing reference. The beacon

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

parameters N_B and T_B are chosen so that there are sufficient beacon slots for a parked slave to synchronize to during a certain time window in an error-prone environment.

When parked, the slave shall receive the beacon parameters through an LMP command. In addition, the timing of the beacon instant is indicated through the offset D_B. As with the SCO logical transport (see 8.8.6.2), two initialization procedures (1 or 2) are used. The master shall use initialization 1 when the MSB of the current master clock CLK_{27} is 0; it shall use initialization 2 when the MSB of the current master clock CLK_{27} is 1. The chosen initialization procedure shall also be carried by an initialization flag in the LMP command. The slave shall apply the initialization method as indicated by the initialization flag irrespective of its clock bit CLK_{27}. The master-to-slave slot positioned at the beacon instant shall be initialized on the slots for which the clock satisfies the applicable equation:

$CLK_{27-1} \bmod T_B = D_B$ for initialization 1

$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_B = D_B$ for initialization 2

This implies that D_B will be even.

After initialization, the clock value $CLK(k + 1)$ for the next beacon instant shall be derived by adding the fixed interval T_B to the clock value of the current beacon instant:

$$CLK(k + 1) = CLK(k) + T_B$$

The beacon train serves four purposes:

a) Transmitting master-to-slave packets that the parked slaves can use for resynchronization
b) Carrying messages to the parked slaves to change the beacon parameters
c) Carrying general broadcast messages to the parked slaves
d) Unparking one or more parked slaves

Since a slave can synchronize to any packet that is preceded by the proper CAC, the packets carried on the beacon slots do not have to contain specific broadcast packets for parked slaves to be able to synchronize; any packet may be used. The only requirement placed on the beacon slots is that there is a master-to-slave transmission present on the hopping sequence associated with the park structure. If there is no information to be sent, NULL packets may be transmitted by the master. If there is broadcast information to be sent to the parked slaves, the first packet of the broadcast message shall be repeated in every beacon slot of the beacon train. However, synchronous traffic in the synchronous reserved slots may interrupt the beacon transmission if it is on the same hopping sequence as the parked slaves. The master shall configure its park beacon structure so that reserved slots of synchronous logical transports do not cause slaves to miss synchronisation on a beacon slot. For example, a master that has active slaves using AHS, and parked slaves using non-AHS shall ensure that the park beacons cannot be interrupted by AHS synchronous reserved slots.

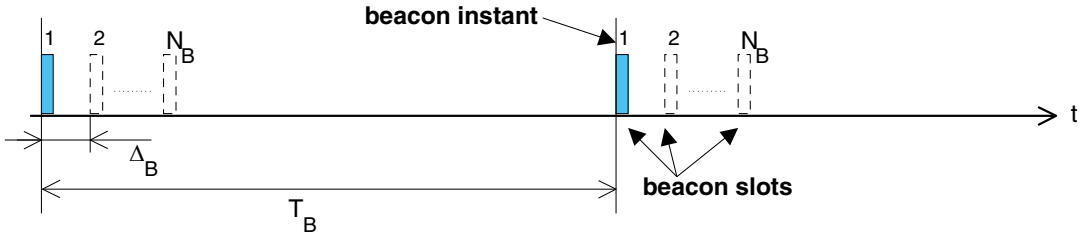


Figure 72—General beacon train format

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The master can place parked slaves in any of the AFH operating modes, but shall ensure that all parked slaves use the same hop sequence. Masters should use AHS(79) or AHS when all the slaves in the piconet are AFH capable.

A master that switches a slave between AFH-enabled, AFH-disabled, or AHS(79) operation shall ensure that the AFH_Instant occurs before transmission of the beacon train using this hop sequence.

The master communicates with parked slaves using broadcast messages. Since these messages can be time-critical, an ongoing repetition train of broadcast message may be prematurely aborted by broadcast information destined to parked slaves in beacon slots and in access windows (see 8.8.9.2).

8.8.9.2 Beacon access window

In addition to the beacon slots, an access window is defined where the parked slaves can send requests to be unparked. To increase reliability, the access window may be repeated M_{access} times ($M_{access} \geq 1$) (see Figure 73). The access window starts a fixed delay D_{access} after the beacon instant. The width of the access window is T_{access}.

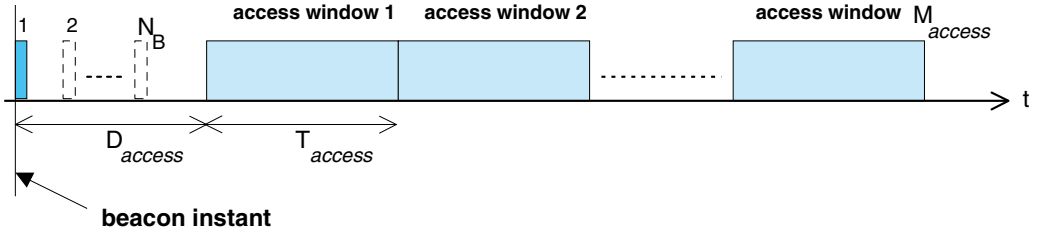


Figure 73—Definition of access window

The access window supports a polling slave access technique. The format of the polling technique is shown in Figure 74. The same TDD structure is used as on the piconet physical channel, i.e., master-to-slave transmission is alternated with slave-to-master transmission. The slave-to-master slot is divided into two half slots of 312.5 µs each. The half slot in which a parked slave is allowed to respond corresponds to its AR_ADDR (see also 8.8.9.6). For counting the half slots to determine the access request slot, the start of the access window is used (see Figure 74). The slave shall send an access request in the proper slave-to-master half slot only if a broadcast packet has been received in the preceding master-to-slave slot. In this way, the master polls the parked slaves.

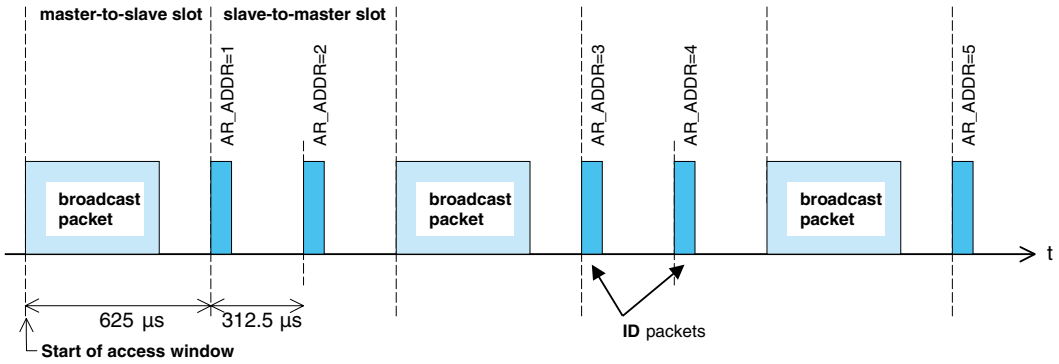


Figure 74—Access procedure applying the polling technique

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The slots of the access window may also be used for traffic on the piconet if required. For example, if a synchronous connection has to be supported, the slots reserved for the synchronous link may carry synchronous information instead of being used for access requests, i.e., if the master-to-slave slot in the access window contains a packet different from a broadcast packet, the following slave-to-master slot shall not be used for slave access requests. If the master transmits a broadcast packet in the access window, then it shall use the hop sequence associated with the park structure. Slots in the access window not affected by piconet traffic may still be used according to the defined access structure (an example is shown in Figure 75), and the access procedure shall be continued as if no interruption had taken place.

When the slave is parked, the master shall indicate what type of access scheme will be used. For the polling scheme, the number of slave-to-master access slots N_{acc_slot} is indicated.

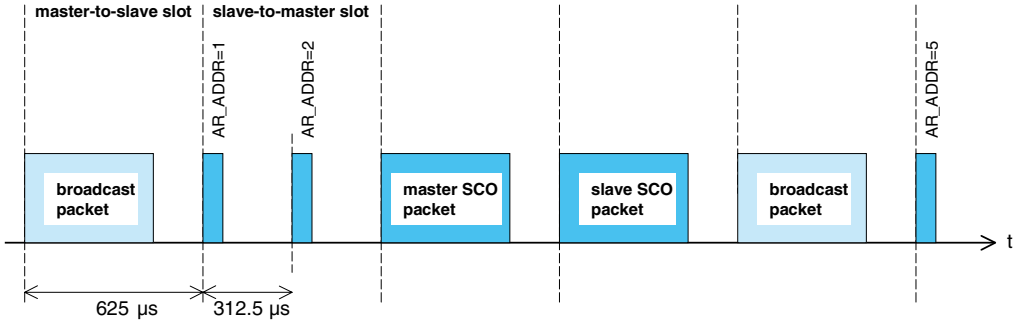


Figure 75—Disturbance of access window by SCO traffic

By default, the access window is always present. However, its activation depends on the master sending broadcast messages to the slave at the appropriate slots in the access window. A flag in a broadcast LMP message within the beacon slots may indicate that the access window(s) belonging to this instant will not be activated. This prevents unnecessary scanning of parked slaves that want to request access.

8.8.9.3 Parked slave synchronization

Parked slaves wake up periodically to resynchronize to the channel. Any packet exchanged on the channel can be used for synchronization. Since master transmission is mandatory on the beacon slots, parked slaves will use the beacon train to resynchronize. A parked slave may wake up at the beacon instant to read the packet sent on the first beacon slot. If this fails, it may retry on the next beacon slot in the beacon train; in total, there are N_B opportunities per beacon instant to resynchronize. During the search, the slave may increase its search window (see also 8.2.2.5.2). The separation between the beacon slots in the beacon train Δ_B shall be chosen so that consecutive search windows will not overlap.

The parked slave may not wake up at every beacon instant. Instead, a sleep interval may be applied that is longer than the beacon interval T_B (see Figure 76). The slave sleep window shall be a multiple N_{B_sleep} of T_B. The precise beacon instant on which the slave may wake up shall be indicated by the master with D_{B_sleep}, which indicates the offset (in multiples of T_B) with respect to the beacon instant ($0 < D_{B\_sleep} < N_{B\_sleep} - 1$). To initialize the wake-up period, the applicable equation shall be used:

$$CLK_{27-1} \bmod (N_{B\_sleep} * T_B) = D_B + D_{B\_sleep} * T_B \text{ for initialization 1}$$

$$(\overline{CLK}_{27}, CLK_{26-1}) \bmod (N_{B\_sleep} * T_B) = D_B + D_{B\_sleep} * T_B \text{ for initialization 2}$$

where initialization 1 shall be chosen by the master if the MSB in the current CLK is 0 and initialization 2 shall be chosen by the master if the MSB in the current CLK is 1.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When the master needs to send broadcast messages to the parked slaves, it may use the beacon slots for these broadcast messages. However, if $N_B < N_{BC}$, the slots following the last beacon slot in the beacon train shall be used for the remaining $N_{BC} - N_B$ broadcast packets. If $N_B > N_{BC}$, the broadcast message shall be repeated on all N_B beacon slots.

A parked slave shall read the broadcast messages sent in the beacon slot(s) in which it wakes up. If the parked slave wakes up, the minimum wake-up activity shall be to read the CAC for resynchronization and the packet header to check for broadcast messages.

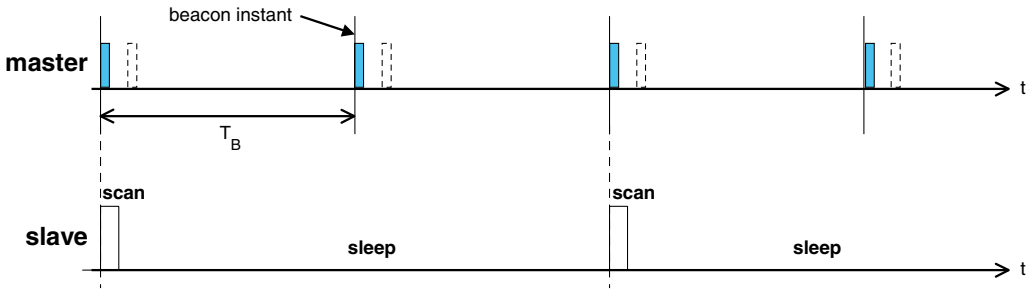


Figure 76—Extended sleep interval of parked slaves

8.8.9.4 Parking

A master can park an active slave through the exchange of LMP commands. Before being put into PARK state, the slave shall be assigned a PM_ADDR and an AR_ADDR. Every parked slave shall have a unique PM_ADDR or a PM_ADDR of 0. The AR_ADDR is not necessarily unique. The beacon parameters shall be given by the master when the slave is parked. The slave shall then give up its LT_ADDR and shall enter PARK state. A master can park only a single slave at a time. The park message is carried with a **DM1** packet and addresses the slave through its LT_ADDR.

8.8.9.5 Master-initiated unparking

The master can unpark a parked slave by sending a dedicated LMP unpark command including the parked slave's address. This message shall be sent in a broadcast packet on the beacon slots. The master shall use either the slave's PM_ADDR or its BD_ADDR. The message also includes the LT_ADDR the slave shall use after it has reentered the piconet. The unpark message may include a number of slave addresses so that multiple slaves may be unparked simultaneously. For each slave, a different LT_ADDR shall be assigned.

After having received the unpark message, the parked slave matching the PM_ADDR or BD_ADDR shall leave the PARK state and enter the CONNECTION state. It shall keep listening to the master until it is addressed by the master through its LT_ADDR. The first packet sent by the master shall be a POLL packet. The return packet in response to the POLL packet confirms that the slave has been unparked. If no response packets from the slave are received for *newconnectionTO* number of slots after the end of beacon repetition period, the master shall unpark the slave again. The master shall use the same LT_ADDR on each unpark attempt until the link supervision timer for that slave has elapsed or the unpark has completed successfully. If the slave does not receive the POLL packet for *newconnectionTO* number of slots after the end of beacon repetition period, it shall return to PARK state, with the same beacon parameters. After confirming that the slave is in the CONNECTION state, the master decides in which mode the slave will continue.

When a device is unparked, the SEQN bit for the link shall be reset to 1 on both the master and the slave (see 8.7.6.2.1).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.8.9.6 Slave-initiated unparking

A slave can request access to the channel through the access window defined in 8.8.9.2. As shown in Figure 74 (in 8.8.9.2), the access window includes several slave-to-master half slots where the slave may send an access request message. The specific half slot in which the slave is allowed to respond corresponds to its AR_ADDR, which it received when it was parked. The order of the half slots (in Figure 74 the AR_ADDR numbers linearly increase from 1 to 5) is not fixed: an LMP command sent in the beacon slots may reconfigure the access window. When a slave desires access to the channel, it shall send an access request message in the proper slave-to-master half slot. The access request message of the slave is the ID packet containing the DAC of the master (which is the CAC without the trailer). The parked slave shall transmit an access request message in the half slot only when, in the preceding master-to-slave slot, a broadcast packet has been received. This broadcast message may contain any kind of broadcast information not necessarily related to the parked slave(s). If no broadcast information is available, a broadcast NULL or broadcast POLL packet may be sent to enable the access window.

After having sent an access request, the parked slave shall listen for an unpark message from the master. As long as no unpark message is received, the slave shall repeat the access requests in the subsequent access windows. After the last access window (there are M_{access} windows in total; see 8.8.9.2), the parked slave shall listen for an additional N_{poll} time slots for an unpark message. If no unpark message is received within N_{poll} slots after the end of the last access window, the slave may return to sleep and retry an access attempt after the next beacon instant.

After having received the unpark message, the parked slave matching the PM_ADDR or BD_ADDR shall leave the PARK state and enter the CONNECTION state. It shall keep listening to the master until it is addressed by the master through its LT_ADDR. The first packet sent by the master shall be a POLL packet. The return packet in response to the POLL packet confirms that the slave has been unparked. After confirming that the slave is in the CONNECTION state, the master decides in which mode the slave will continue. If no response packet from the slave is received for *newconnectionTO* number of slots after N_{poll} slots after the end of the last access window, the master shall send the unpark message to the slave again. If the slave does not receive the POLL packet for *newconnectionTO* number of slots after N_{poll} slots after the end of the last access window, it shall return to PARK state with the same beacon parameters.

When a device is unparked, the SEQN bit for the link shall be reset to 1 on both the master and the slave (see 8.7.6.2.1).

8.8.9.7 Broadcast scan window

In the beacon train, the master can support broadcast messages to the parked slaves. However, it may extend its broadcast capacity by indicating to the parked slaves that more broadcast information is following after the beacon train. This is achieved by an LMP command ordering the parked slaves (as well as the active slaves) to listen to the channel for broadcast messages during a limited time window. This time window starts at the beacon instant and continues for the period indicated in the LMP command sent in the beacon train.

8.8.9.8 Polling in the PARK state

In the PARK state, parked slaves may send access requests in the access window provided a broadcast packet is received in the preceding master-to-slave slot. Slaves in the CONNECTION state shall not send in the slave-to-master slots following the broadcast packet since they are allowed to send only if addressed specifically.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.9 Audio

On the air-interface, either a 64 kb/s log pulse code modulation (PCM) format (A-law or μ-law) may be used or a 64 kb/s continuous variable slope delta (CVSD) modulation may be used. The latter format applies an adaptive delta modulation algorithm with syllabic companding.

The voice coding on the line interface is designed to have a quality equal to or better than the quality of 64 kb/s log PCM.

Table 30 summarizes the voice coding schemes supported on the air interface. The appropriate voice coding scheme is selected after negotiations between the LMs.

Table 30—Voice coding schemes supported on the air interface

Voice codecs	
Linear	CVSD
8-bit logarithmic	A-law
	μ-law

8.9.1 Log PCM coder decoder (CODEC)

Since the synchronous logical transports on the air interface can support a 64 kb/s information stream, a 64 kb/s log PCM traffic can be used for transmission. Either A-law or μ-law compression may be applied. In the event that the line interface uses A-law and the air interface uses μ-law or vice versa, a conversion from A-law to μ-law or vice versa shall be performed. The compression method shall follow ITU-T Recommendation G.711.

8.9.2 CVSD CODEC

A more robust format for voice-over-air interface is delta modulation. This modulation scheme follows the waveform where the output bits indicate whether the prediction value is smaller or larger than the input waveform. To reduce slope overload effects, syllabic companding is applied: the step size is adapted according to the average signal slope. The input to the CVSD encoder shall be 64 Ksample/s linear PCM (typically 16 bits, but actual value is implementation specific). Block diagrams of the CVSD encoder and CVSD decoder are shown in Figure 77, Figure 78, and Figure 79. The system shall be clocked at 64 kHz.

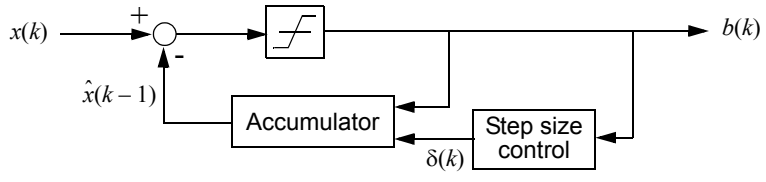


Figure 77—Block diagram of CVSD encoder with syllabic companding

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

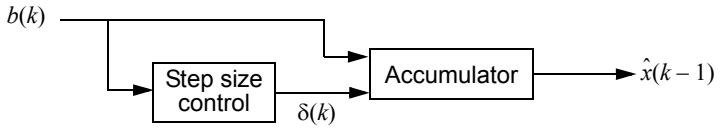


Figure 78—Block diagram of CVSD decoder with syllabic companding

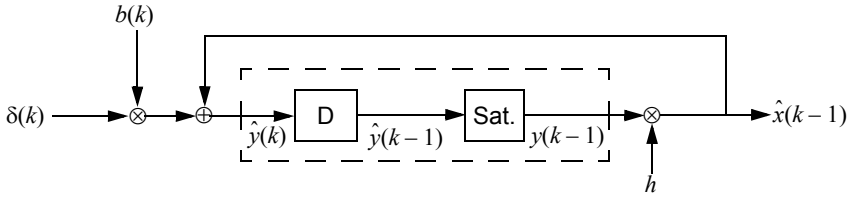


Figure 79—Accumulator procedure

Let $\mathrm{sgn}(x) = 1$ for $x \geq 0$; otherwise, $\mathrm{sgn}(x) = -1$. On air, these numbers shall be represented by the sign bit, i.e., negative numbers are mapped on 1, and positive numbers are mapped on 0.

Denote the CVSD encoder output bit $b(k)$, the encoder input $x(k)$, the accumulator contents $y(k)$, and the step size $\delta(k)$. Furthermore, let h be the decay factor for the accumulator, let β denote the decay factor for the step size, and, let α be the syllabic companding parameter. The last parameter monitors the slope by considering the K most recent output bits.

Let

$$\hat{x}(k) = hy(k). \tag{13}$$

Then, the CVSD encoder internal state shall be updated according to the following set of equations:

$$b(k) = \mathrm{sgn}\{x(k) - \hat{x}(k-1)\}, \tag{14}$$

$$\alpha = \begin{cases} 1, & \text{if } J \text{ bits in the last } K \text{ output bits are equal,} \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

$$\delta(k) = \begin{cases} \min\{\delta(k-1) + \delta_{\min}, \delta_{\max}\}, & \alpha = 1, \\ \max\{\beta\delta(k-1), \delta_{\min}\}, & \alpha = 0, \end{cases} \tag{16}$$

$$y(k) = \begin{cases} \min\{\hat{y}(k), y_{\max}\}, & \hat{y}(k) \geq 0. \\ \max\{\hat{y}(k), y_{\min}\}, & \hat{y}(k) < 0. \end{cases} \tag{17}$$

where

$$\hat{y}(k) = \hat{x}(k-1) + b(k)\delta(k). \tag{18}$$

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

In these equations, δ_{min} and δ_{max} are the minimum and maximum step sizes, and, y_{min} and y_{max} are the accumulator's negative and positive saturation values, respectively. Over air, the bits shall be sent in the same order they are generated by the CVSD encoder.

For a 64 kb/s CVSD, the parameters as shown in Table 31 shall be used. The numbers are based on a 16-bit signed number output from the accumulator. These values result in a time constant of 0.5 ms for the accumulator decay and a time constant of 16 ms for the step size decay.

Table 31—CVSD parameter values[a]

Parameter	Value
h	$1 - \frac{1}{32}$
b	$1 - \frac{1}{1024}$
J	4
K	4
δ_{min}	10
δ_{max}	1280
y_{min}	-2^{15} or $-2^{15}+1$
y_{max}	$2^{15}-1$

[a]The values are based on a 16-bit signed number output from the accumulator.

8.9.3 Error handling

In the **DV**, **HV3**, **EV3**, and **EV5** packets, the voice is not protected by FEC. The quality of the voice in an error-prone environment then depends on the robustness of the voice coding scheme and, in the case of eSCO, the retransmission scheme. CVSD, in particular, is rather insensitive to random bit errors, which are experienced as white background noise. When a packet is rejected because the CAC is incorrect, the HEC test was unsuccessful, or the CRC has failed, measures have to be taken to "fill" in the lost speech segment.

The voice payload in the **HV2** and **EV4** packets is protected by a 2/3 rate FEC. For errors that are detected, but cannot be corrected, the receiver should try to minimize the audible effects. For instance, from the 15-bit FEC segment with uncorrected errors, the 10-bit information part as found before the FEC decoder should be used. The **HV1** packet is protected by a 3-bit repetition FEC. For this code, the decoding scheme will always assume zero or 1-bit errors. Thus, there exist no detectable, but uncorrectable, error events for **HV1** packets.

8.9.4 General audio requirements

8.9.4.1 Signal levels

For A-law and μ-law log PCM-encoded signals, the requirements on signal levels shall follow the ITU-T Recommendation G.711.

Full swing at the 16-bit linear PCM interface to the CVSD encoder is defined to be 3 dBm0.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.9.4.2 CVSD audio quality

The requirements for audio quality are put on the TX side. The 64 Ksample/s linear PCM input signal should have negligible spectral power density above 4 kHz. The power spectral density in the 4–32 kHz band of the decoded signal at the 64 Ksample/s linear PCM output should be more than 20 dB below the maximum in the 0–4 kHz range.

8.10 General audio recommendations

8.10.1 Maximum sound pressure

It is the sole responsibility of each manufacturer to design its audio products in a safe way with regard to injury to the human ear. This standard does not specify maximum sound pressure from an audio device.

8.10.2 Other telephony network requirements

It is the sole responsibility of each manufacturer to design the audio product so that it meets the regulatory requirements of all telephony networks to which it may be connected.

8.10.3 Audio levels

Audio levels shall be calculated as send loudness rating (SLR) and receive loudness rating (RLR). The calculation methods are specified in ITU-T Recommendation P.79 [B14].

The physical test setup for handsets and headsets is described in ITU-T Recommendations P.51 [B12] and P.57 [B13].

The physical test setup for speakerphones and vehicle handsfree systems is specified in ITU-T Recommendation P.34 [B11].

A general equation for computation of loudness rating (LR) for telephone sets is given by ITU-T Recommendations P.79 and is given by

$$R = -\frac{10}{m}\log 10\left(\sum_{i=N_1}^{N_2} 10^{ms_i - w_i/10}\right), \tag{19}$$

where

m is a constant (~ 0.2);
w_i is weighting coefficient (different for the various LRs);
S_i is the sensitivity at frequency F_i, of the electro-acoustic path;
N_1,N_2 are consecutive filter bank numbers (Art. Index: 200–4000 Hz).

Equation (19) is used for calculating the SLR according to Figure 80 and RLR according to Figure 81. When calculating LRs, one must include only the parts of the frequency band where an actual signal transmission can occur in order to ensure that the additive property of LRs is retained. Therefore, ITU-T Recommendation P.79 uses only the 200–4000 Hz frequency band in LR computations.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.10.4 Microphone path

The SLR measurement model is shown in Figure 80.

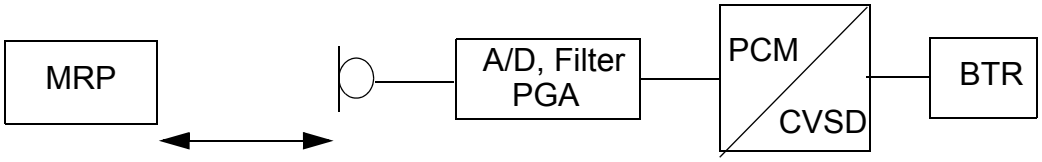


Figure 80—SLR measurement setup

8.10.5 Loudspeaker path

The RLR measurement model is shown in Figure 81.

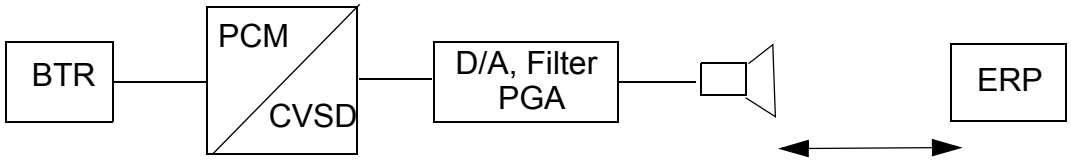


Figure 81—RLR measurement setup

8.10.6 Voice interface

The voice interface should follow in the first place the ITU-T Recommendations P.79 [B14], which specifies the LRs for telephone sets. These recommendations give general guidelines and specific algorithms used for calculating the LRs of the audio signal with respect to ear reference point (ERP).

For voice interfaces to the different cellular system terminals, loudness and frequency recommendations based on the cellular standards should be used. Since this standard is based on the Bluetooth specification, any recommendations for Bluetooth devices may also be applied to IEEE 802.15.1-2005 devices. For example, GSM 03.50 gives recommendation for both the LRs and frequency mask for a GSM terminal interconnection with IEEE 802.15.1-2005.

a) The output of the CVSD decoder are 16-bit linear PCM digital samples, at a sampling frequency of 8 ksample/s. IEEE 802.15.1-2005 also supports 8-bit log PCM samples of A-law and μ-law type. The sound pressure at the ERP for a given 16-bit CVSD sample should follow the sound pressure level given in the cellular standard specification.
b) A maximum sound pressure that can be represented by a 16-bit linear PCM sample at the output of the CVSD decoder should be specified according to the LR in ITU Recommendation P.79 and at the programmable gain amplifier (PGA) value of 0 dB. PGAs are used to control the audio level at the terminals by the user. For conversion between various PCM representations (A-law, μ-law, and linear PCM), ITU-T Recommendations G.711, G.712 [B9], and G.714 [B10] give guidelines and PCM value relationships. Zero-code suppression based on ITU-T Recommendation G.711 is also recommended to avoid network mismatches.

8.10.7 Frequency mask

For interfacing a device to a digital cellular mobile terminal, a compliance of the CVSD decoder signal to the frequency mask given in the cellular standard is recommended to guarantee correct function of the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

speech coders. A recommendation for a frequency mask is given in Table 32. Figure 82 shows a plot of the frequency mask for IEEE 802.15.1-2005 (solid line). The GSM frequency mask (dotted line) is shown in Figure 82 for comparison.

Table 32—Recommended frequency mask for IEEE 802.15.1-2005

Frequency (Hz)	Upper limit (dB)	Lower limit (dB)
50	–20	—
300	4	–12
1000	4	–9
2000	4	–9
3000	4	–9
3400	4	–12
4000	0	—

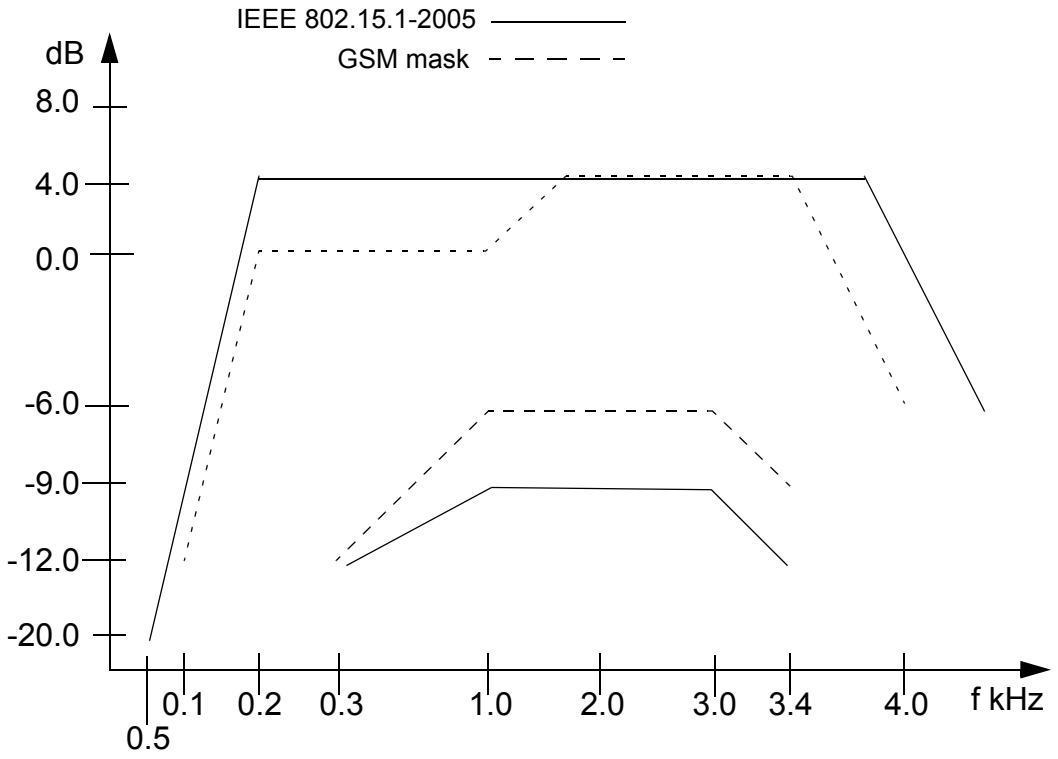


Figure 82—Plot of recommended frequency mask for IEEE 802.15.1-2005 with GSM send frequency mask given for comparison

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.11 Timers

This subclause contains a list of BB timers related to inactivity timeout defined in this standard. Definitions and default values of the timers are listed in 8.11.1 through 8.11.5.

All timer values are given in slots.

8.11.1 inquiryTO

The *inquiryTO* timer defines the number of slots the **inquiry** substate will last. The timer value may be changed by the host. HCI provides a command to change the timer value.

8.11.2 pageTO

The *pageTO* timer defines the number of slots the **page** substate can last before a response is received. The timer value may be changed by the host. HCI provides a command to change the timer value.

8.11.3 pagerespTO

In the slave, the *pageresp*TO timer defines the number of slots the slave awaits the master's response (FHS packet) after sending the page acknowledgment ID packet. In the master, the *pageresp*TO timer defines the number of slots the master should wait for the FHS packet acknowledgment before returning to the **page** substate. Both master and slave devices should use the same value for this timeout to ensure common page/scan intervals after reaching *pageresp*TO.

The value of the *pageresp*TO timer is 8 slots.

8.11.4 newconnectionTO

Every time a new connection is started through paging, scanning, role switching, or unparking, the master sends a POLL packet as the first packet in the new connection. Transmission and acknowledgment of this POLL packet are used to confirm the new connection. If the POLL packet is not received by the slave or the response packet is not received by the master for *newconnectionTO* number of slots, both the master and the slave will return to the previous substate.

The value of the *newconnectionTO* timer is 32 slots.

8.11.5 supervisionTO

The *supervisionTO* timer is used by both the master and slave to monitor link loss. If a device does not receive any packets that pass the HEC check and have the proper LT_ADDR for a period of *supervisionTO*, it will reset the link. The supervision timer keeps running in HOLD mode, SNIFF mode, and PARK state.

The *supervisionTO* value may be changed by the host. HCI provides a command to change the timer value. At the BB level, a default value that is equivalent to 20 s will be used.

8.12 Recommendations for AFH operation in PARK, HOLD, and SNIFF

The three possible AFH operation modes for an AFH-capable slave in PARK state, HOLD mode, and SNIFF mode are the same three AFH operation modes used during CONNECTION state:

— Enabled (using the same AHS as slaves in the CONNECTION state)
— Enabled [using AHS(79)]

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— Disabled

The master may place an AFH-capable slave in any of the three AFH operating modes.

8.12.1 Operation at the master

A master that has one or more slaves in PARK state, HOLD mode, or SNIFF mode and decides to update them simultaneously shall schedule an AFH_Instant for a time that allows it to update all these slaves (as well as its active slaves) with the new adaptation.

A master that has multiple slaves with nonoverlapping “wake” times (e.g., slaves in SNIFF mode with different timing parameters) may keep them enabled on the same adaptation provided that its scheduling of the AFH_Instant allows enough time to update them all.

This timing is summarized in Figure 83. In this example, the master decides that a hop sequence adaptation is required. However, it cannot schedule an AFH_Instant until it has informed its active slave, its slave in HOLD mode, and its slave in SNIFF mode and had time to unpark its parked slaves.

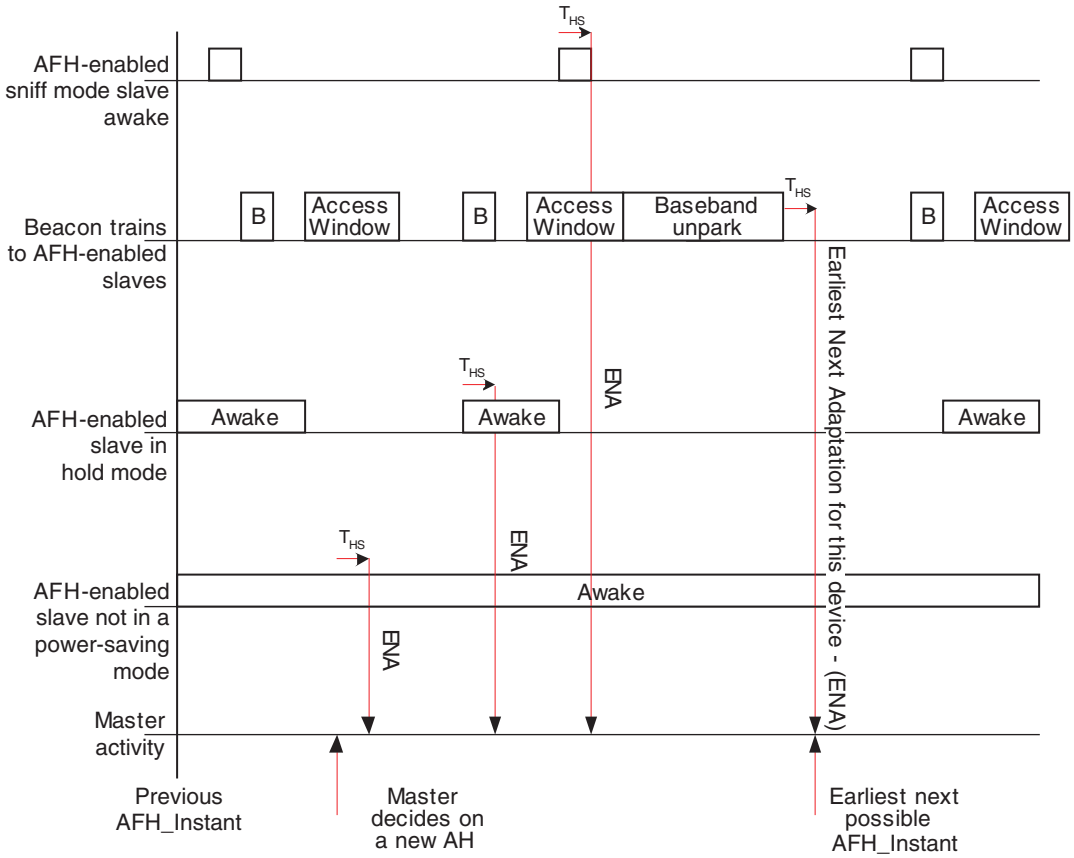


Figure 83—Timing constraint on AFH_Instant with slaves in PARK, HOLD, and SNIFF

8.12.2 Operation in PARK state

A slave that is in the PARK state cannot send or receive any AFH LMP messages (see 9.3.5.2). Once the slave has left the PARK state, the master may subsequently update the slave’s adaptation.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

8.12.3 AFH operation in SNIFF mode

Once a slave has been placed in SNIFF mode, the master may periodically change its AHS without taking the slave out of SNIFF mode.

8.12.4 AFH operation in HOLD mode

A slave that is in HOLD mode cannot send or receive any LMP messages. Once the slave has left HOLD mode, the master may subsequently update the slave's adaptation.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9. Link Manager Protocol (LMP)

This clause describes the LMP, which is used for link setup and control. The signals are interpreted and filtered out by the LM on the receiving side and are not propagated to higher layers.

The LMP is used to control and negotiate all aspects of the operation of the IEEE 802.15.1-2005 connection between two devices. This includes the setup and control of logical transports and logical links and the control of physical links.

The LMP is used to communicate between the LMs on the two devices that are connected by the ACL logical transport. All LMP messages shall apply solely to the physical link and associated logical links and logical transports between the sending and receiving devices.

The protocol is made up of a series of messages that shall be transferred over the ACL-C logical link on the default ACL logical transport between two devices. LMP messages shall be interpreted and acted upon by the LM and shall not be directly propagated to higher protocol layers.

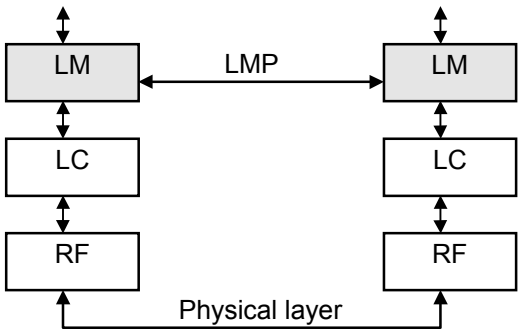


Figure 84—LMP signalling

9.1 General rules

9.1.1 Message transport

LMP messages shall be exchanged over the ACL-C logical link that is carried on the default ACL logical transport (see 8.4.4). The ACL-C logical link is distinguished from the ACL-U (which carries L2CAP and user data) by the LLID field carried in the payload header of variable-length packets (see 8.6.6.2).

The ACL-C has a higher priority than other traffic (see 8.5.5).

The error detection and correction capabilities of the BB ACL logical transport are generally sufficient for the requirements of the LMP. Therefore, LMP messages do not contain any additional error detection information beyond what can be realized by means of sanity checks performed on the contents of LMP messages. Any such checks and protections to overcome undetected errors in LMP messages are an implementation matter.

9.1.2 Synchronization

This subclause is informative and explains why many of the LMP message sequences are defined as they are.

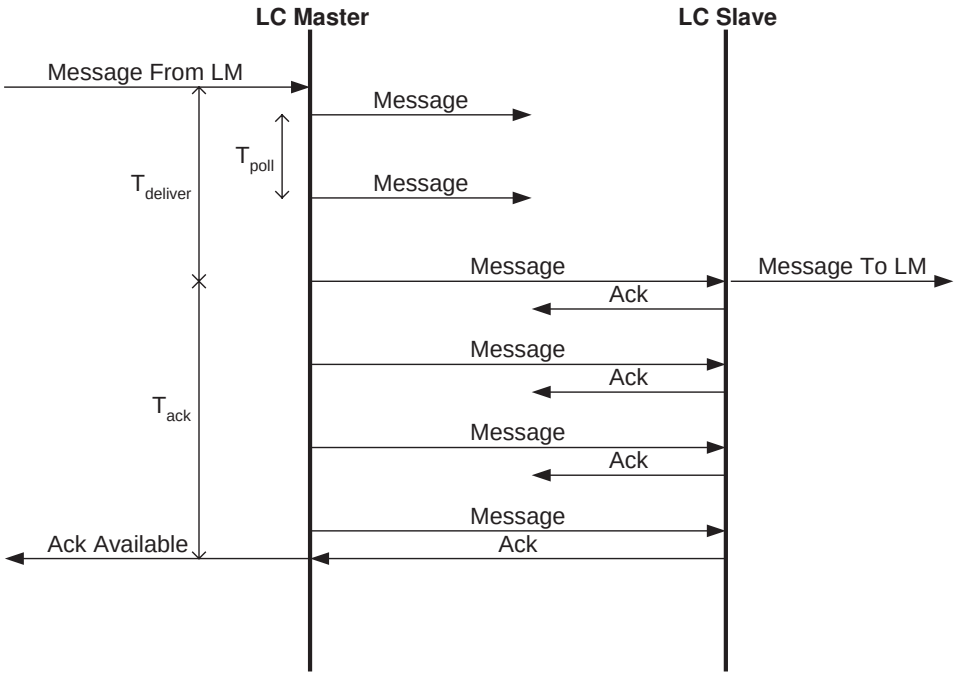
Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

LMP messages are carried on the ACL-C logical link, which does not guarantee a time to deliver or time to acknowledge packets. LMP procedures take account of this when synchronizing state changes in the two devices. For example, criteria are defined that specify when a LT_ADDR may be reused after it becomes available due to a device leaving the piconet or entering the PARK state. Other LMP procedures, such as HOLD or role switch, include the CLK as a parameter in order to define a fixed synchronization point. The transitions into and out of SNIFF mode are protected with a transition mode.

The link controller normally attempts to communicate with each slave no less often than every T_{poll} slots (see 9.3.1.8) based on the T_{poll} for that slave.

Figure 85 illustrates the fundamental problem. It shows the transmission of a packet from the master to the slave in conditions of heavy interference for illustrative purposes. It is obvious that neither side can determine the value of either $T_{deliver}$ or T_{ack}. It is, therefore, not possible to use simple messages to identify uniquely the instant at which a state change occurs in the other device.



Note that the diagram shows the limiting case where the master transmits the message at intervals of T_{poll}. In the case of heavy interference, improved performance is gained by transmitting more often.

Figure 85—Transmission of message from master to slave

9.1.3 Packet format

Each PDU is assigned either a 7-bit or a 15-bit opcode used to uniquely identify different types of PDUs (see Table 67 in 9.4). The first 7 bits of the opcode and a transaction ID are located in the first byte of the payload body. If the initial 7 bits of the opcode have one of the special escape values 124–127, then an additional byte of opcode is located in the second byte of the payload, and the combination uniquely identifies the PDU.

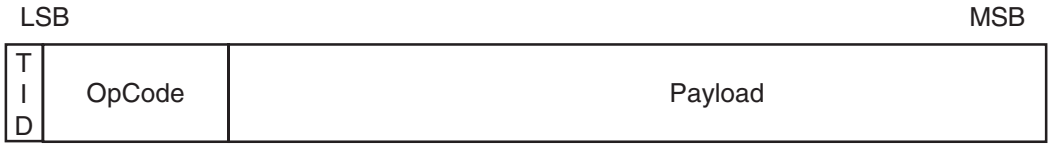
The FLOW bit in the packet header is always 1 and is ignored on reception.

If the PDU contains one or more parameters, these are placed in the payload starting immediately after the opcode, i.e., at byte 2 if the PDU has a 7-bit opcode or byte 3 if the PDU has a 15-bit opcode. The number of

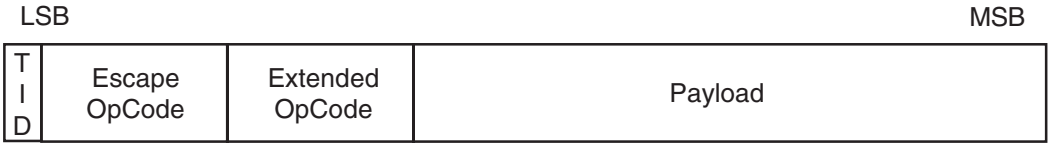
Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

bytes used depends on the length of the parameters. All parameters have a little-endian format, i.e., the least significant byte is transferred first.

LMP messages shall be transmitted using **DM1** packets; however, if an **HV1** SCO link is in use and the length of the payload is less than 9 bytes, then **DV** packets may be used.



LMP PDU with 7 bit opCode



LMP PDU with 15 bit opCode

Figure 86—Payload body when LMP PDUs are sent

9.1.4 Transactions

The LMP operates in terms of transactions. A transaction is a connected set of message exchanges that achieve a particular purpose. All PDUs that form part of the same transaction shall have the same value for the transaction ID that is stored as part of the first byte of the opcode (see 9.1.3).

The transaction ID is in the LSB. It shall be 0 if the PDU forms part of a transaction that was initiated by the master and 1 if the transaction was initiated by the slave.

Each sequence described in 9.3 shall be defined as a transaction. For pairing (see 9.3.2.2) and encryption (see 9.3.2.5), all sequences belonging to each subclause shall be counted as one transaction and shall use the same transaction ID. For connection establishment (see 9.3.1.1), LMP_host_connection_req and the response with LMP_accepted or LMP_not_accepted shall form one transaction and have the transaction ID of 0. LMP_setup_complete is a stand-alone PDU, which forms a transaction by itself.

For error handling (see 9.1.5), the PDU to be rejected and LMP_not_accepted or LMP_not_accepted_ext shall form a single transaction.

9.1.4.1 LMP response timeout

The time between receiving a BB packet carrying an LMP PDU and sending a BB packet carrying a valid response PDU, according to the procedure rules in 9.3, shall be less than the LMP response timeout. The value of this timeout is 30 s. Note that the LMP response timeout is applied not only to sequences described in 9.3, but also to the series of the sequences defined as the transactions in 9.3. It shall also be applied to the series of LMP transactions that take place during a period when traffic on the ACL-U logical link is disabled for the duration of the series of LMP transactions, e.g., during the enabling of encryption.

The LMP response timeout shall restart each time an LMP PDU that requires a reply is queued for transmission by the BB.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.1.5 Error handling

If the LM receives a PDU with an unrecognized opcode, it shall respond with an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *unknown LMP PDU*. The opcode parameter that is echoed back is the unrecognized opcode.

If the LM receives a PDU with invalid parameters, it shall respond with an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *invalid LMP parameters*.

If the maximum response time (see 9.1.4) is exceeded or if a link loss is detected (see 8.3.1), the party that waits for the response shall conclude that the procedure has terminated unsuccessfully.

Erroneous LMP messages can be caused by errors on the channel or systematic errors at the TX side. To detect the latter case, the LM should monitor the number of erroneous messages and disconnect if it exceeds a threshold, which is implementation dependent.

When the LM receives a PDU that is not allowed and the PDU normally expects a PDU reply, e.g., LMP_host_connection_req or LMP_unit_key, the LM shall return an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *PDU not allowed*. If the PDU normally does not expect a reply, e.g., LMP_sres or LMP_temp_key, the PDU will be ignored.

The reception of an optional PDU that is not supported shall be handled in one of two ways: If the LM simply does not know the opcode (e.g., it was added at a later version of this standard), it shall respond with an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *unknown LMP PDU*. If the LM recognises the PDU as optional, but unsupported, then it shall reply with an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *unsupported LMP feature* if the PDU would normally generate a reply; otherwise, no reply is generated.

9.1.5.1 Transaction collision resolution

Since LMP PDUs are not interpreted in real time, collision situations can occur where both LMs initiate the same procedure and both procedures cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *LMP error transaction collision*. The master-initiated procedure shall then be completed.

Collision situations can also occur where both LMs initiate different procedures and both procedures cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending an LMP_not_accepted or LMP_not_accepted_ext PDU with the error code *LMP error different transaction collision*. The master- initiated procedure shall then be completed.

9.1.6 Procedure rules

Each procedure is described and depicted with a sequence diagram. The symbols in Figure 87 are used in the sequence diagrams.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

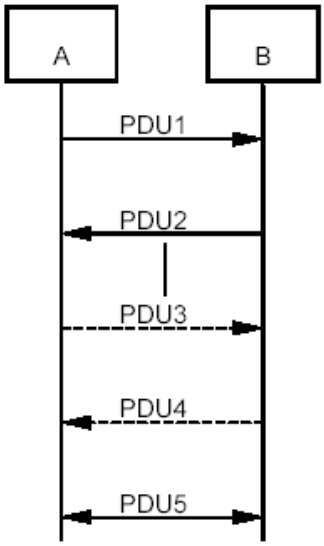


Figure 87—Symbols used in sequence diagrams

PDU1 is a PDU sent from A to B. PDU2 is a PDU sent from B to A. PDU3 is a PDU that is optionally sent from A to B. PDU4 is a PDU that is optionally sent from B to A. PDU5 is a PDU sent from either A or B. A vertical line indicates that more PDUs can optionally be sent.

9.1.7 General response messages

The PDUs LMP_accepted, LMP_accepted_ext, LMP_not_accepted, and LMP_not_accepted_ext are used as response messages to other PDUs in a number of different procedures. LMP_accepted or LMP_accepted_ext includes the opcode of the message that is accepted. LMP_not_accepted or LMP_not_accepted_ext includes the opcode of the message that is not accepted and the error code indicating why it is not accepted. See Table 33

LMP_accepted_ext and LMP_not_accepted_ext shall be used when the opcode is 15 bits in length. LMP_accepted and LMP_not_accepted shall be used when the opcode is 7 bits in length.

Table 33—General response messages

M/O[a]	PDU	Contents
M	LMP_accepted	Opcode
M	LMP_not_accepted	Opcode Error code
O	LMP_accepted_ext	Escape opcode Extended opcode
O	LMP_not_accepted_ext	Escape opcode Extended opcode Error code

[a]M = mandatory; O = optional.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.1.8 LMP message constraints

This subclause is informative.

— No LMP message shall exceed the maximum payload length of a single **DM1** packet, i.e., 17 bytes in length (see 8.6.5.4.1).
— All LM messages are of fixed length apart from those sent using broadcast in PARK state.
— The LMP version number is not be used to indicate the presence or absence of functionality.

9.2 Device features

Each PDU is either mandatory or optional as defined by the M/O columns in the tables of 9.3. An M in this column shall indicate that support for the feature is mandatory. An O in this column shall indicate that support for the PDU is optional, and it shall be followed by the number(s) of the feature(s) involved (in parentheses).

All features added after IEEE Std 802.15.1-2002 have associated LMP feature bits. Support of these features may be made “mandatory” by the qualification process, but the LM still considers them to be optional since it must interoperate with older devices that do not support them.

The LM does not need to be able to transmit a PDU that is optional. Support of optional PDUs is indicated by a device’s features mask. The features mask can be read (see 9.3.3.4). An LM shall not send or be sent any PDU that is incompatible with the features signaled in its features mask or the features mask of its peer, as detailed in 9.2.1.

9.2.1 Feature definitions

Table 34 provides summary definitions of the features provided by LMP and references the subclauses containing more detailed definitions. Some features have only local meaning and do not imply support for any additional LMP PDUs or sequences. Although local features have no meaning for the remore LM, they are still included in the feature definitions because they are meaningful to the local host. In systems implementing HCI, the host may read the LM features using the HCI_Read_Local_Supported_Features command.

Table 34—Feature definitions

Feature	Definition
Extended features	This feature indicates whether the device is able to support the extended features mask using the LMP sequences defined in 9.3.3.4.
Timing accuracy	This feature indicates whether the LM supports requests for timing accuracy using the sequence defined in 9.3.3.1.
Enhanced inquiry scan	This feature indicates whether the device is capable of supporting the enhanced inquiry scan mechanism as defined in 8.8.4.1. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.
Interlaced inquiry scan	This feature indicates whether the device is capable of supporting the interlaced inquiry scan mechanism as defined in 8.8.4.1. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.
Interlaced page scan	This feature indicates whether the device is capable of supporting the interlaced page scan mechanism as defined in 8.8.3.1. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 34—Feature definitions *(continued)*

Feature	Definition
RSSI with inquiry results	This feature indicates whether the device is capable of reporting the RSSI with inquiry results as defined in 8.8.4.2. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.
Paging parameter negotiation	This feature indicates whether the LM is capable of supporting the signaling of changes in the paging scheme as defined in 9.3.1.9.
3-slot packets	This feature indicates whether the device supports the transmission and reception of both **DM3** and **DH3** packets for the transport of traffic on the ACL-U logical link.
5-slot packets	This feature indicates whether the device supports the transmission and reception of both **DM5** and **DH5** packets for the transport of traffic on the ACL-U logical link.
Flow control lag	This is defined as the total amount of ACL-U data that can be sent following the receipt of a valid payload header with the payload header FLOW bit set to 0 and is in units of 256 bytes. See further details in 8.6.6.2.
AFH-capable slave	This feature indicates whether the device is able to support the AFH mechanism as a slave as defined in 8.2.3 using the LMP sequences defined in 9.3.1.4.
AFH classification slave	This feature indicates whether the device is able to support the AFH classification mechanism as a slave as defined in 8.8.6.8 using the LMP sequences defined in 9.3.1.5.
AFH-capable master	This indicates whether the device is able to support the AFH mechanism as a master as defined in 8.2.3 using the LMP sequences defined in 9.3.1.4.
AFH classification master	This feature indicate whether the device is able to support the AFH classification mechanism as a master as defined in 8.8.6.8 using the LMP sequences defined in 9.3.1.5.
Power control	This feature indicates whether the device is capable of adjusting its transmission power. This feature indicates the ability to receive the LMP_incr_power and LMP_decr_power PDUs and transmit the LMP_max_power and LMP_min_power PDUs, using the sequences defined in 9.3.1.3. These sequences may be used even if the remote device does not support the power control feature, as long as it supports the power control requests feature.
Power control requests	This feature indicates whether the device is capable of determining if the transmit power level of the other device should be adjusted and will send the LMP_incr_ power and LMP_decr_power PDUs to request the adjustments. It indicates the ability to receive the LMP_max_power and LMP_min_power PDUs, using the sequences in 9.3.1.3. These sequences may be used even if the remote device does not support the RSSI feature, as long as it supports the power control feature.
CQDDR	This feature indicates whether the LM is capable of recommending a packet type (or types) depending on the channel quality using the LMP sequences defined in 9.3.1.7.
Broadcast encryption	This feature indicates whether the device is capable of supporting broadcast encryption as defined in 13.4.2 and also the sequences defined in 9.3.2.5.5 and 9.3.2.4. NOTE—Devices compliant to IEEE Std 802.15.1-2002 may support broadcast encryption even though this feature bit is not set.
Encryption	This feature indicates whether the device supports the encryption of packet contents using the sequence defined in 9.3.2.5.
Slot offset	This feature indicates whether the LM supports the transfer of the slot offset using the sequence defined in 9.3.4.1.
Role switch	This feature indicates whether the device supports the change of master and slave roles as defined by 8.8.6.5 using the sequence defined in 9.3.4.2.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 34—Feature definitions *(continued)*

Feature	Definition
HOLD mode	This feature indicates whether the device is able to support HOLD mode as defined in 8.8.8 using the LMP sequences defined in 9.3.5.1.
SNIFF mode	This feature indicates whether the device is able to support SNIFF mode as defined in 8.8.7 using the LMP sequences defined in 9.3.5.3.
PARK state	This feature indicates whether the device is able to support PARK state as defined in 8.8.9 using the LMP sequences defined in 9.3.5.2.
SCO link	This feature indicates whether the device is able to support the SCO logical transport as defined in 8.4.3, the **HV1** packet defined in 8.6.5.2.1, and the **DV** packet defined in 8.6.5.2.4 using the LMP sequence in 9.3.6.1.
HV2 packets	This feature indicates whether the device is capable of supporting the **HV2** packet type as defined in 8.6.5.2.2 on the SCO logical transport.
HV3 packets	This feature indicates whether the device is capable of supporting the **HV3** packet type as defined in 8.6.5.2.3 on the SCO logical transport.
μ-law log synchronous data	This feature indicates whether the device is capable of supporting μ-law log format data as defined in 8.9.1 on the SCO and eSCO logical transports.
A-law log synchronous data	This feature indicates whether the device is capable of supporting A-law log format data as defined in 8.9.1 on the SCO and eSCO logical transports.
CVSD synchronous data	This feature indicates whether the device is capable of supporting CVSD format data as defined in 8.9.2 on the SCO and eSCO logical transports.
Transparent synchronous data	This feature indicates whether the device is capable of supporting transparent synchronous data as defined in 8.6.4.3 on the SCO and eSCO logical transports.
Extended SCO link	This feature indicates whether the device is able to support the eSCO logical transport as defined in 8.5.4 and the **EV3** packet defined in 8.6.5.3.1 using the LMP sequences defined in 9.3.6.2.
EV4 packets	This feature indicates whether the device is capable of supporting the **EV4** packet type defined in 8.6.5.3.2 on the eSCO logical transport.
EV5 packets	This feature indicates whether the device is capable of supporting the **EV5** packet type defined in 8.6.5.3.3 on the eSCO logical transport.

9.2.2 Features mask definition

The features are represented as a bit mask when they are transferred in LMP messages. For each feature, a single bit is specified, which shall be set to 1 if the feature is supported and set to 0 otherwise. The single exception is the flow control lag, which is coded as a 3-bit field with the LSB in byte 2 bit 4 and the MSB in byte 2 bit 6. All unknown or unassigned feature bits shall be set to 0. See Table 35.

Table 35—Features mask definition

Number	Supported feature	Byte	Bit
0	3-slot packets	0	0
1	5-slot packets	0	1
2	Encryption	0	2

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 35—Features mask definition *(continued)*

Number	Supported feature	Byte	Bit
3	Slot offset	0	3
4	Timing accuracy	0	4
5	Role switch	0	5
6	HOLD mode	0	6
7	SNIFF mode	0	7
8	PARK state	1	0
9	Power control requests	1	1
10	CQDDR	1	2
11	SCO link	1	3
12	**HV2** packets	1	4
13	**HV3** packets	1	5
14	μ-law log synchronous data	1	6
15	A-law log synchronous data	1	7
16	CVSD synchronous data	2	0
17	Paging parameter negotiation	2	1
18	Power control	2	2
19	Transparent synchronous data	2	3
20	Flow control lag(LSB)	2	4
21	Flow control lag(middle bit)	2	5
22	Flow control lag(MSB)	2	6
23	Broadcast encryption	2	7
24	Reserved	3	0
25	Reserved	3	1
26	Reserved	3	2
27	Enhanced inquiry scan	3	3
28	Interlaced inquiry scan	3	4
29	Interlaced page scan	3	5
30	RSSI with inquiry results	3	6
31	Extended SCO link (**EV3** packets)	3	7
32	**EV4** packets	4	0
33	**EV5** packets	4	1
34	Reserved	4	2
35	AFH-capable slave	4	3

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 35—Features mask definition *(continued)*

Number	Supported feature	Byte	Bit
36	AFH classification slave	4	4
37	Reserved	4	5
38	Reserved	4	6
43	AFH-capable master	5	3
44	AFH classification master	5	4
63	Extended features	7	7

9.2.3 LM interoperability policy

LMs of any version will interoperate using the lowest common subset of functionality by reading the LMP features mask (defined in Table 35).

An optional LMP PDU shall be sent to a device only if the corresponding feature bit is set in its features mask. The exception to this are certain PDUs (see 9.3.1.1) that can be sent before the features mask is requested.

NOTE—A later version device with a restricted feature set is indistinguishable from an earlier version device with the same features.[11]

9.3 Procedure rules

This subclause describes the rules for carrying out LMP procedures.

9.3.1 Connection control

This subclause describes the LMP procedures for controlling connections, including connection establishment, detach, and power control within a connection.

9.3.1.1 Connection establishment

After the paging procedure, LMP procedures for clock offset request, LMP version, supported features, name request, and detach may be initiated.

An LM connection may be established during a device discovery phase in order to retrieve information such as the LM version, supported features, and the device's name. Such a connection does not involve higher layers and, therefore, does not need to proceed to an LM_host_connection_req PDU. After the desired information has been retrieved, an LM_detach PDU is sent. Figure 88 illustrates this transaction.

A typical connection establishment that is initiated by host request is shown in Figure 89.

[11]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement this standard.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

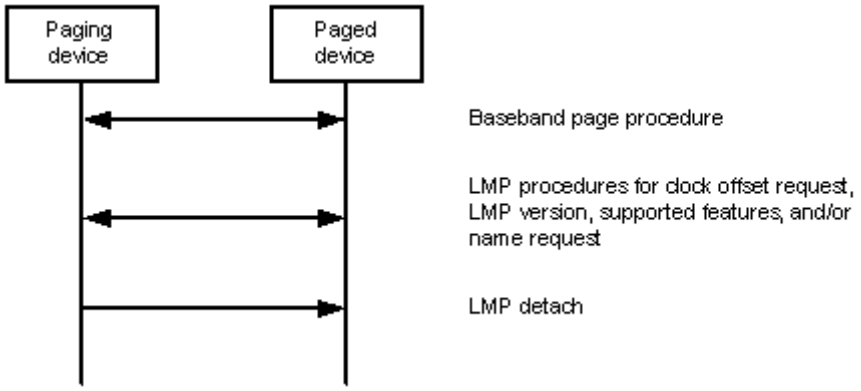


Figure 88—Connection establishment without host request

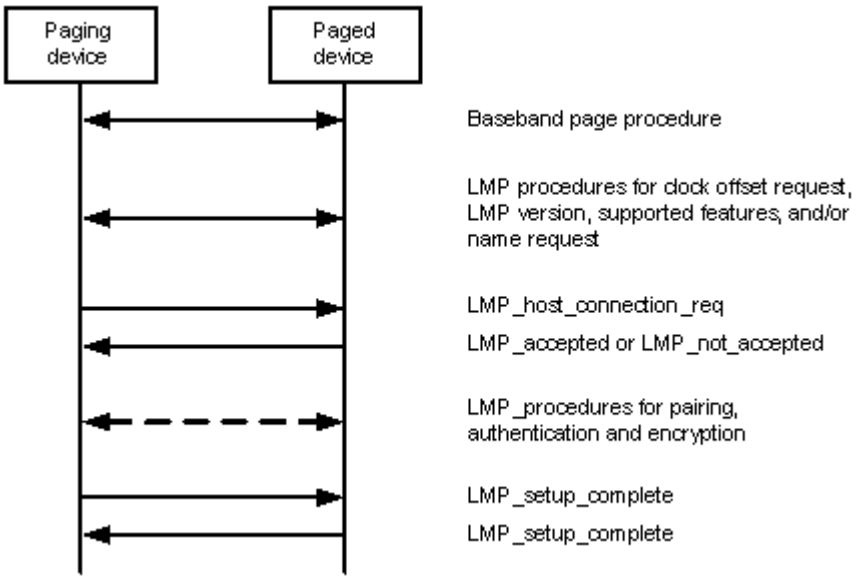


Figure 89—Connection establishment with host request

When the paging device wishes to create a connection involving layers above LM, it sends an LMP_host_connection_req PDU. When the other side receives this message, the host is informed about the incoming connection. The remote device can accept or reject the connection request by sending an LMP_accepted PDU or an LMP_not_accepted PDU. Alternatively, if the slave needs a role switch (see 9.3.4.2), it sends an LMP_slot_offset PDU and LMP_switch_req PDU after it has received an LMP_host_connection_req PDU. If the role switch fails, the LM shall continue with the creation of the connection unless this cannot be supported due to limited resources. In this case, the connection shall be terminated with an LMP_detach PDU with error code *other end terminated connection: low resources*. When the role switch has been successfully completed, the old slave will reply with an LMP_accepted PDU or an LMP_not_accepted PDU to the LMP_host_connection_req PDU (with the transaction ID set to 0).

If the paging device receives an LMP_not_accepted PDU in response to an LMP_host_connection_req PDU, it shall immediately disconnect the link using the mechanism described in 9.3.1.2.

If the LMP_host_connection_req PDU is accepted, LMP security procedures (pairing, authentication, and encryption) may be invoked. When a device is not going to initiate any more security procedures during connection establishment, it sends an LMP_setup_complete PDU. When both devices have sent LMP_setup_complete PDUs, the traffic can be transferred on the ACL-U logical transport. See Table 36.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 36—PDUs used for connection establishment

M/O	PDU	Contents
M	LMP_host_connection_req	—
M	LMP_setup_complete	—

9.3.1.2 Detach

The connection between two devices may be detached anytime by the master or the slave. An error code parameter is included in the message to inform the other party of why the connection is detached. See Table 37.

Table 37—PDU used for detach

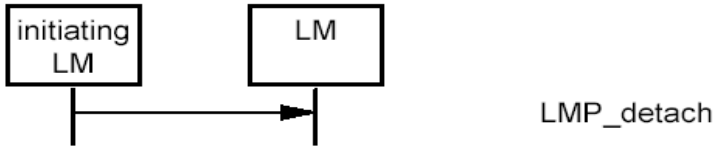
M/O	PDU	Contents
M	LMP_detach	Error code

The initiating LM shall pause traffic on the ACL-U logical link (see 8.5.3.1). The initiating LM then queues the LMP_detach PDU for transmission, and it shall start a timer for $6*T_{poll}$ slots where T_{poll} is the poll interval for the connection. If the initiating LM receives the BB acknowledgment before the timer expires, it starts a timer for $3*T_{poll}$ slots. When this timer expires and if the initiating LM is the master, the LT_ADDR(s) may be reused immediately. If the initial timer expires, then the initiating LM drops the link and starts a timer for $T_{linksupervisiontimeout}$ slots after which the LT_ADDR(s) may be reused if the initiating LM is the master.

When the receiving LM receives the LMP_detach PDU, it shall start a timer for $6*T_{poll}$ slots if it is the master and $3*T_{poll}$ if it is the slave. On timer expiration, the link shall be detached and, if the receiving LM is the master, the LT_ADDR(s) may be reused immediately. If the receiver never receives the LMP_detach PDU, then a link supervision timeout will occur, the link will be detached, and the LT_ADDR may be reused immediately.

If, at any time during this or any other LMP sequence, the link supervision timeout expires, then the link shall be terminated immediately, and the LT_ADDR(S) may be reused immediately.

If the connection is in HOLD mode, the initiating LM shall wait for the HOLD mode to end before initiating the procedure defined above (in this subclause). If the connection is in SNIFF mode or PARK state, the initiating LM shall perform the procedure to exit SNIFF mode or PARK state before initiating the procedure defined above. If the procedure to exit SNIFF mode or PARK state does not complete within the LMP response timeout (30 s), the procedure defined above shall be initiated anyway. See Sequence 1.



Sequence 1: Connection closed by sending LMP_detach.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.1.3 Power control

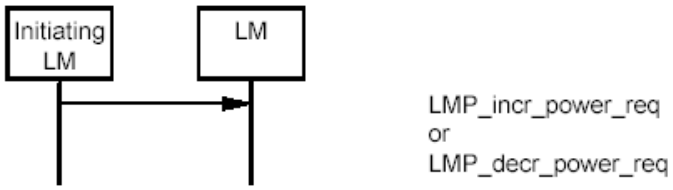
If the received signal characteristics differs too much from the preferred value of a device, it may request an increase or a decrease of the other device's TX power. The power adjustment requests may be made at any ime following a successful BB paging procedure.

If a device does not support power control requests, this is indicated in the supported features list, and thus no power control requests shall be sent after the supported features response has been processed. Prior to this time, a power control adjustment might be sent. If the recipient does not support power control, it is allowed to send an LMP_max_power PDU in response to an LMP_incr_power_req PDU and an LMP_min_power PDU in response to an LMP_decr_power_req PDU. Another possibility is to send an LMP_not_accepted PDU with the error code *unsupported LMP feature*.

Upon receipt of an LMP_incr_power_req PDU or LMP_decr_power_req PDU, the output power shall be increased or decreased one step unless this would take power above the device's maximum power level or below its minimum power level. See Clause 7 for the definition of the step size. The TX power is a property of the physical link and affects all logical transports carried over the physical link. Power control requests carried over the default ACL-C logical link shall affect only the physical link associated with the default ACL-C logical link: they shall not affect the power level used on the physical links to other slaves. See Table 38 and Sequence 2.

Table 38—PDUs used for power control

M/O	PDU	Contents
O(9)	LMP_incr_power_req	For future use (1 byte)
O(9)	LMP_decr_power_req	For future use (1 byte)
O(18)	LMP_max_power	—
O(18)	LMP_min_power	—



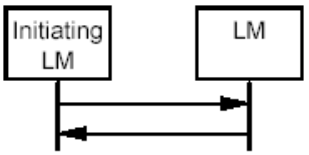
Sequence 2: A device requests a change of the other device's TX power.

If the receiver of an LMP_incr_power_req PDU is at maximum power, an LMP_max_power PDU shall be returned. The device shall request an increase again only after having requested a decrease at least once. If the receiver of an LMP_decr_power_req PDU is at minimum power, then an LMP_min_power PDU shall be returned, and the device shall request a decrease only after having requested an increase at least once. See Sequence 3 and Sequence 4.

One byte is reserved in an LMP_incr/decr_power_req PDU for future use. The parameter value shall be 0x00 and ignored upon receipt.

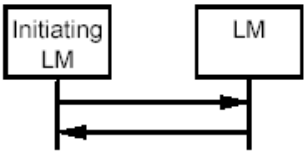
Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



LMP_incr_power_req
LMP_max_power

Sequence 3: The TX power cannot be increased.



LMP_decr_power_req
LMP_min_power

Sequence 4: The TX power cannot be decreased.

9.3.1.4 Adaptive frequency hopping (AFH)

AFH is used to improve the performance of physical links in the presence of interference as well as reducing the interference caused by physical links on other devices in the ISM band. AFH shall be used only during the CONNECTION state. See Table 39.

Table 39—PDUs used for AFH

M/O	PDU	Contents
O(35) Rx O(43) Tx	LMP_set_AFH	AFH_Instant, AFH_Mode, AFH_Channel_Map

The LMP_set_AFH PDU contains three parameters: AFH_Instant, AFH_Mode, and AFH_Channel_Map. The AFH_Instant parameter specifies the instant at which the hopset switch will become effective. This is specified as the value of the master's clock (CLK), which is available to both devices. The AFH instant is chosen by the master and shall be an even value at least $6*T_{poll}$ or 96 slots (whichever is greater) in the future, where T_{poll} is at least the longest poll interval for all AFH-enabled physical links. The AFH instant shall be within 12 hr of the current clock value. The AFH_Mode parameter specifies whether AFH shall be enabled or disabled. The AFH_Channel_Map parameter specifies the set of channels that shall be used if AFH is enabled.

When the LMP_set_AFH PDU is received, the AFH instant shall be compared with the current CLK. If it is in the past, then the AFH instant has passed, and the slave shall immediately configure the hop selection kernel (see 8.2.6.3) with the new AFH_Mode and AFH_Channel_Map parameters specified in the LMP_set_AFH PDU. If it is in the future, then a timer shall be started to expire at the AFH instant. When this timer expires, it shall configure the hop selection kernel with the new AFH_Mode and AFH_Channel_Map parameters.

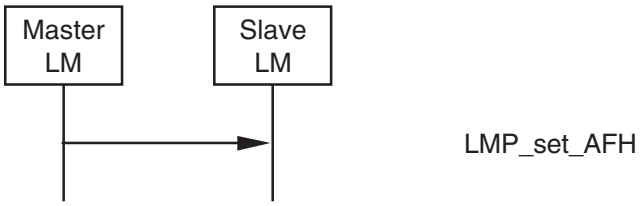
The master shall not send a new LMP_set_AFH PDU to a slave until it has received the BB acknowledgment for any previous LMP_set_AFH PDU addressed to that slave and the instant has passed.

Role switch while AFH is enabled shall follow the procedures define by 8.8.6.5.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.1.4.1 Master enables AFH

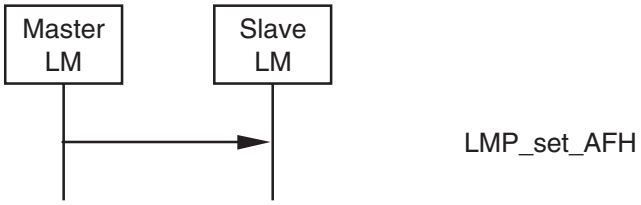
Prior to enabling AFH, the master LM shall pause traffic on the ACL-U logical link (see 8.5.3.1). The master shall then enable AFH on a physical link by sending the LMP_set_AFH PDU with the AFH_Mode parameter set to AFH_enabled, the AFH_channel_map parameter containing the set of used and unused channels, and the AFH_instant parameter set. The LM shall not calculate the AFH instant until after traffic on the ACL-U logical link has been stopped. The master considers the physical link to be AFH enabled once the BB acknowledgment has been received and the AFH instant has passed. Once the BB acknowledgment has been received, the master shall restart transmission on the ACL-U logical link. See Sequence 5.



Sequence 5: Master enables AFH.

9.3.1.4.2 Master disables AFH

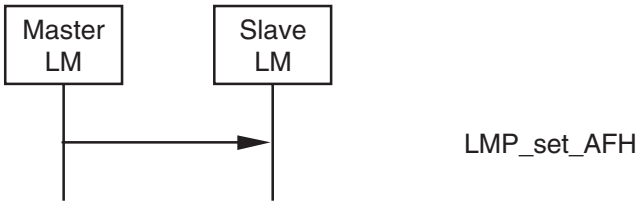
Prior to disabling AFH, the master LM shall pause traffic on the ACL-U logical link (8.5.3.1). The master shall then disable AFH operation on a physical link by sending the LMP_set_AFH PDU with the AFH_Mode parameter set to AFH_disabled and the AFH_Instant parameter set. The AFH_Channel_Map parameter is not valid when AFH mode is AFH disabled. The LM shall not calculate the AFH instant until after traffic on the ACL-U logical link has been stopped. The master considers the physical link to have entered AFH-disabled operation once the BB acknowledgment has been received and the AFH _instant has passed. Once the BB acknowledgment has been received, the master shall restart transmission on the ACL-U logical link. See Sequence 6.



Sequence 6: Master disables AFH.

9.3.1.4.3 Master updates AFH

A master shall update the AFH parameters on a physical link by sending the LMP_set_AFH PDU with the AFH_Mode parameter set to AFH_enabled, the AFH_Instant parameter set, and a new AFH_Channel_Map parameter set. The master shall consider the slave to have the updated AFH parameters once the BB acknowledgment has been received and the AFH instant has passed. See Sequence 7.



Sequence 7: Master updates AFH.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.1.4.4 AFH operation in PARK, HOLD, and SNIFF

A slave in the PARK state, HOLD mode, or SNIFF mode shall retain the AFH_Mode and AFH_Channel_Map parameters prior to entering those modes. A master may change the AFH mode while a slave is in SNIFF mode.

A master that receives a request from an AFH-enabled slave to enter PARK state, HOLD mode, or SNIFF mode and decides to operate the slave using a different hop sequence shall respond with an LMP_set_AFH PDU specifying the new hop sequence.

The master continues with the LMP signalling, for park, hold, or sniff initiation, once the BB acknowledgment for the LMP_set_AFH PDU has been received. Optionally, the master may delay the continuation of this LMP signalling until after the instant. An AFH-capable slave device shall support both of these cases.

A master that receives a request from an AFH-enabled slave to enter PARK state, HOLD mode, or SNIFF mode and decides to not change the slave's hop sequence shall respond exactly as it would do without AFH. In this case, AFH operation has no effect on the LMP signalling.

9.3.1.5 Channel classification

A master may request channel classification information from a slave that is AFH enabled.

A slave that supports the AFH classification slave feature shall perform channel classification and reporting according to its AFH reporting mode. The master shall control the AFH reporting mode using the LMP_channel_classification_req PDU. The slave shall report its channel classification using the LMP_channel_classification PDU.

The slave shall report pairs of channels as good, bad, or unknown. See Table 68 for the detailed format of the AFH_Channel_Classification parameter. When one channel in the nth channel pair is good and the other channel is unknown, the nth channel pair shall be reported as good. When one channel in the nth channel pair is bad and the other is unknown, the nth channel pair shall be reported as bad. It is implementation dependent what to report when one channel in a channel pair is good and the other is bad. See Table 40.

Table 40—PDUs used for channel classification reporting

M/O	PDU	Contents
O(36) Rx O(44) Tx	LMP_channel_classification_req	AFH_Reporting_Mode, AFH_Min_Interval, AFH_Max_Interval
O(36) Tx O(44) Rx	LMP_channel_classification	AFH_Channel_Classification

The LMP_channel_classification_req PDU contains three parameters: AFH_Reporting_Mode, AFH_Min_Interval, and AFH_Max_Interval. The AFH_Min_Interval parameter defines the minimum amount of time from the last LMP_channel_classification command that was sent before the next LMP_channel_classification PDU may be sent. The AFH_Max_Interval parameter defines the maximum amount of time between the change in the radio environment being detected by a slave and its generation of an LMP_channel_classification PDU. The AFH maximum interval shall be equal to or larger than the AFH minimum interval.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The AFH_Reporting_Mode parameter shall determine if the slave is in the AFH_reporting_enabled or AFH_reporting_disabled state. The default state, prior to receipt of any LMP_channel_classification_req PDUs, shall be AFH_reporting_disabled. In the AFH_reporting_disabled state, the slave shall not generate any channel classification reports.

The AFH_reporting_mode parameter is implicitly set to the AFH_reporting_disabled state when any of the following occur:

— Establishment of a connection at the BB level
— Master-slave role switch
— Entry to PARK state operation
— Entry to HOLD mode

The AFH_reporting_mode parameter is implicitly restored to its former value when any of the following occur:

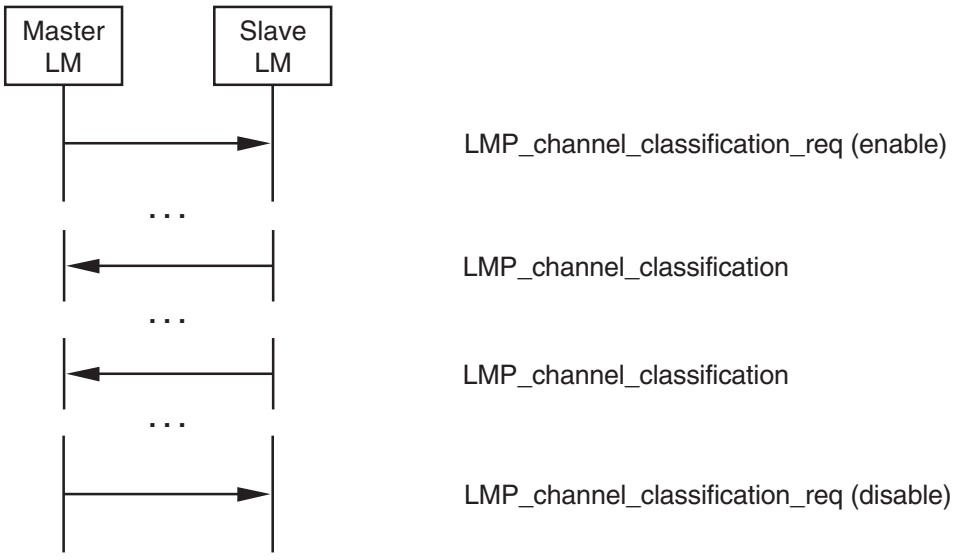
— Exit from PARK state operation
— Exit from HOLD mode
— Failure of master-slave role switch

9.3.1.5.1 Channel classification reporting enabling and disabling

A master enables slave channel classification reporting by sending the LMP_channel_classification_req PDU with the AFH_Reporting_Mode parameter set to AFH_reporting_enabled.

When a slave has had classification reporting enabled by the master, it shall send the LMP_channel_classification PDU according to the information in the latest LMP_channel_classification_req PDU. The LMP_channel_classification PDU shall not be sent if there has been no change in the slave's channel classification.

A master disables slave channel classification reporting by sending the LMP_channel_classification_req PDU with the AFH_Reporting_Mode parameter set to AFH_reporting_disabled. See Sequence 8.



Sequence 8: Channel classification reporting.

Copyright © 2005 IEEE. All rights reserved.

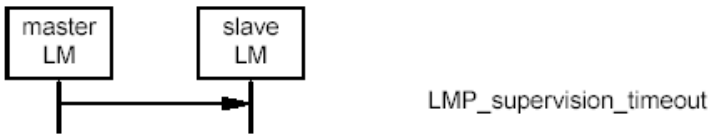
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.1.6 Link supervision

Each physical link has a timer that is used for link supervision. This timer is used to detect physical link loss caused by devices moving out of range or being blocked by interference, a device's power-down, or other similar failure cases. Link supervision is specified in 8.3.1. See Table 41 and Sequence 9.

Table 41—PDU used to set the supervision timeout

M/O	PDU	Contents
M	LMP_supervision_timeout	Supervision timeout



Sequence 9: Setting the link supervision timeout.

9.3.1.7 Channel quality-driven data rate (CQDDR) change

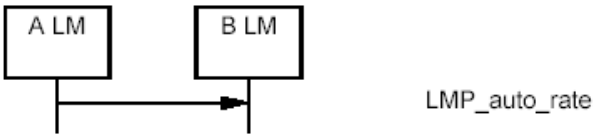
The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput. Device A sends the LMP_auto_rate PDU once to notify device B to enable this feature. Once enabled, device B may change the packet type(s) that device A transmits by sending the LMP_preferred_rate PDU. This PDU has a parameter that determines the preferred coding (with or without 2/3FEC) and optionally the preferred size in slots of the packets. Device A is not required to change to the packet type specified by this parameter. Device A shall not send a packet that is larger than maximum slots (see 9.3.1.10) even if the preferred size is greater than this value.

These PDUs may be sent at any time after connection setup is completed. See Table 42, Sequence 10, and Sequence 11.

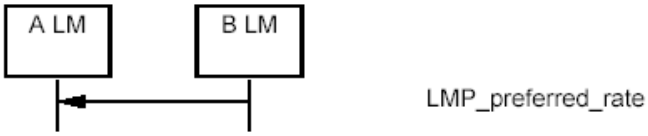
Table 42—PDUs used for quality-driven change of data rate

M/O	PDU	Contents
O(10)	LMP_auto_rate	—
O(10)	LMP_preferred_rate	Data rate

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 10: Device A notifies device B to enable CQDDR.



Sequence 11: Device B sends device A a preferred packet type.

9.3.1.8 Quality of service (QoS)

The LM provides QoS capabilities. A poll interval, T_{poll}, which is defined as the maximum time between transmissions from the master to a particular slave on the ACL logical transport, is used to support bandwidth allocation and latency control (see 8.8.6.1 for details). The poll interval is guaranteed in the active and SNIFF modes except when there are collisions with page, page scan, inquiry, and inquiry scan; during time-critical LMP sequences in the current piconet and any other piconets in which the device is a member; and during critical BB sequences (such as the page response, initial CONNECTION state until the first POLL, and master-slave switch). These PDUs maybe sent at anytime after connection setup is completed.

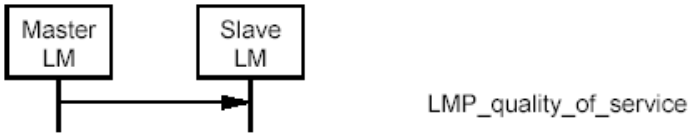
Master and slave negotiate the number of repetitions for broadcast packets (N_{BC}) (see 8.7.6.5). See Table 43.

Table 43— PDUs used for QoS

M/O	PDU	Contents
M	LMP_quality_of_service	Poll interval N_{BC}
M	LMP_quality_of_service_req	Poll interval N_{BC}

9.3.1.8.1 Master notifies slave of the QoS

The master notifies the slave of the new poll interval and N_{BC} by sending the LMP_quality_of_service PDU. The slave cannot reject the notification. See Sequence 12.



Sequence 12: Master notifies slave of QoS.

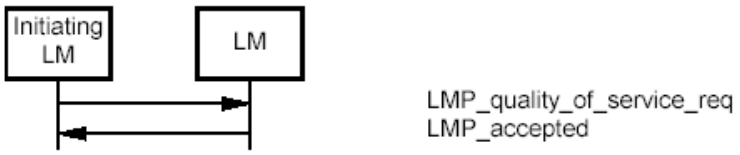
Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

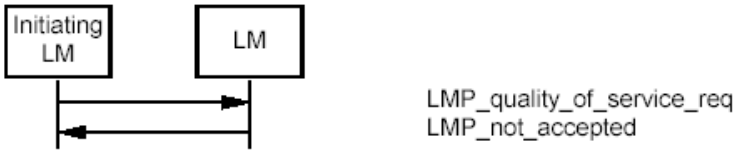
9.3.1.8.2 Device requests new QoS

Either the master or the slave may request a new poll interval and N_{BC} by sending an LMP_quality_of_service_req PDU. The parameter N_{BC} is meaningful only when it is sent by a master to a slave. For transmission of LMP_quality_of_service_req PDUs from a slave, this parameter shall be ignored by the master. The request can be accepted or rejected. This allows the master and slave to dynamically negotiate the QoS as needed.

The selected poll interval by the slave shall be less than or equal to the specified access latency for the outgoing traffic of the ACL link (see 14.5.3). See Sequence 13 and Sequence 14.



Sequence 13: Device accepts new QoS.



Sequence 14: Device rejects new QoS.

9.3.1.9 Paging scheme parameters

LMP provides a means to negotiate the paging scheme parameters that are used the next time a device is paged. See Table 44.

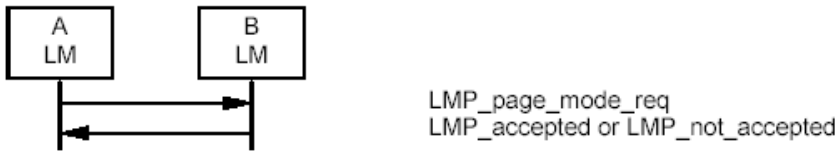
Table 44—PDUs used to request paging scheme

M/O	PDU	Contents
O(17)	LMP_page_mode_req	Paging scheme Paging scheme settings
O(17)	LMP_page_scan_mode_req	Paging scheme Paging scheme settings

9.3.1.9.1 Page mode

This procedure is initiated from device A and negotiates the paging scheme used when device A pages device B. Device A proposes a paging scheme, including the parameters for this scheme, and device B can accept or reject. On rejection, the old setting will not be changed. A request to switch to a reserved paging scheme shall be rejected. See Sequence 15.

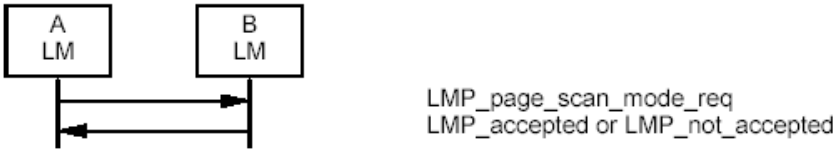
Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 15: Negotiation for page mode.

9.3.1.9.2 Page scan mode

This procedure is initiated from device A and negotiates the paging scheme and paging scheme settings used when device B pages device A. Device A proposes a paging scheme and paging scheme settings, and device B may accept or reject. On rejection, the old setting is not changed. A request specifying the mandatory scheme shall be accepted. A request specifying a nonmandatory scheme shall be rejected. This procedure should be used when device A changes its paging scheme settings. A slave should also send this message to the master after connection establishment to inform the master of the slave's current paging scheme and paging scheme settings. See Sequence 16.



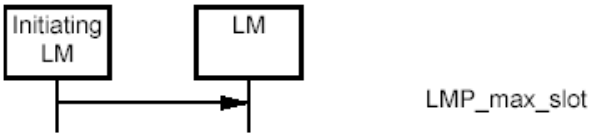
Sequence 16: Negotiation for page scan mode.

9.3.1.10 Control of multislot packets

The number of consecutive slots used by a device on an ACL-U logical link can be limited. It does not affect traffic on the eSCO links where the packet sizes are defined as part of link setup. A device allows the remote device to use a maximum number of slots by sending the LMP_max_slot PDU providing the maximum slots as a parameter. Each device can request to use a maximal number of slots by sending the LMP_max_slot_req PDU providing the maximum slots as a parameter. After a new connection, as a result of page, page scan, role switch, or unpark, the default value is 1 slot. These PDUs can be sent at any time after connection setup is completed. See Table 45, Sequence 17, Sequence 18, and Sequence 19.

Table 45—PDUs used to control the use of multislot packets

M/O	PDU	Contents
M	LMP_max_slot	Maximum slots
M	LMP_max_slot_req	Maximum slots



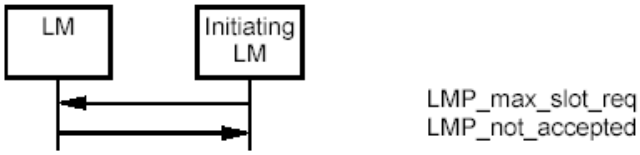
Sequence 17: Device allows remote device to use a maximum number of slots.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 18: Device requests a maximum number of slots. Remote device accepts.



Sequence 19: Device requests a maximum number of slots. Remote device rejects.

9.3.2 Security

9.3.2.1 Authentication

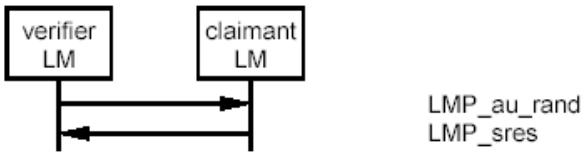
The authentication procedure is based on a challenge-response scheme as described in 13.5. The verifier sends an LMP_au_rand PDU that contains a random number (the challenge) to the claimant. The claimant calculates a response that is a function of this challenge, the claimant's BD_ADDR, and a secret key. The response is sent back to the verifier, which checks if the response was correct or not. The response shall be calculated as described in 13.6.1. A successful calculation of the authentication response requires that two devices share a secret key. This key is created as described in 9.3.2.2. Both the master and the slave can be verifiers. See Table 46.

Table 46—PDUs used for authentication

M/O	PDU	Contents
M	LMP_au_rand	Random number
M	LMP_sres	Authentication response

9.3.2.1.1 Claimant has link key

If the claimant has a link key associated with the verifier, it shall calculate the response and send it to the verifier with an LMP_sres PDU. The verifier checks the response. If the response is not correct, the verifier can end the connection by sending an LMP_detach PDU with the error code *authentication failure* (see 9.3.1.2). See Sequence 20.



Sequence 20: Authentication. Claimant has link key.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Upon reception of an LMP_au_rand PDU, an LM shall reply with an LMP_sres PDU before initiating its own authentication.

NOTE—There can be concurrent requests caused by the master and slave simultaneously initiating an authentication. The procedure in 9.1.5.1 assures that devices will not have different authenticated cyphering offset (ACO) (see 13.6.1) when they calculate the encryption key.

9.3.2.1.2 Claimant has no link key

If the claimant does not have a link key associated with the verifier, it shall send an LMP_not_accepted PDU with the error code *key missing* after receiving an LMP_au_rand PDU. See Sequence 21.



Sequence 21: Authentication fails. Claimant has no link key.

9.3.2.1.3 Repeated attempts

The scheme described in 13.5.1 shall be applied when an authentication fails. This will prevent an intruder from trying a large number of keys in a relatively short time.

9.3.2.2 Pairing

When two devices do not have a common link key, an initialization key (K_{init}) shall be created based on a personal identification number (PIN), a random number, and a BD_ADDR. K_{init} shall be created as specified in 13.6.3. When both devices have calculated K_{init}, the link key shall be created, and a mutual authentication is performed. The pairing procedure starts with a device sending an LMP_in_rand PDU; this device is referred to as the *initiating LM* or *initiator* in 9.3.2.2.1 and 9.3.2.2.5. The other device is referred to as the *responding LM* or *responder*. The PDUs used in the pairing procedure are listed in Table 47.

Table 47—PDUs used for pairing

M/O	PDU	Contents
M	LMP_in_rand	Random number
M	LMP_au_rand	Random number
M	LMP_sres	Authentication response
M	LMP_comb_key	Random number
M	LMP_unit_key	Key

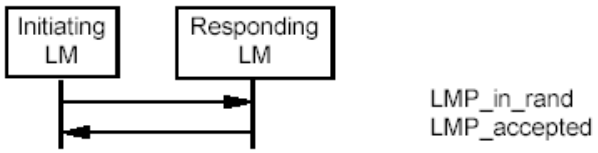
All sequences described in 9.3, including the mutual authentication after the link key has been created, shall form a single transaction. The transaction ID from the first LMP_in_rand PDU shall be used for all subsequent sequences.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.2.2.1 Responder accepts pairing

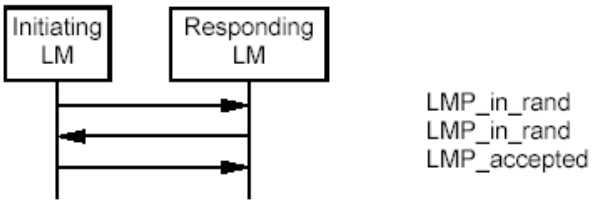
When the initiator sends an LMP_in_rand PDU, the responder shall reply with an LMP_accepted PDU. Both devices shall then calculate K_{init} based on the BD_ADDR of the responder, and the procedure continues with creation of the link key (see 9.3.2.2.4). See Sequence 22.



Sequence 22: Pairing accepted. Responder has a variable PIN, and initiator has a variable or fixed PIN.

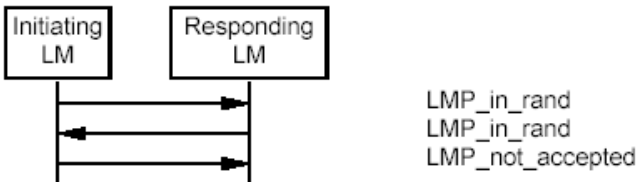
9.3.2.2.2 Responder has a fixed PIN

If the responder has a fixed PIN and accepts pairing, it shall generate a new random number and send it back in an LMP_in_rand PDU. If the initiator has a variable PIN, it shall accept the LMP_in_rand PDU and shall respond with an LMP_accepted PDU. Both sides shall then calculate K_{init} based on the last IN_RAND and the BD_ADDR of the initiator. The procedure continues with creation of the link key (see 9.3.2.2.4). See Sequence 23.



Sequence 23: Responder has a fixed PIN, and initiator has a variable PIN.

If the responder has a fixed PIN and the initiator also has a fixed PIN, the second LMP_in_rand PDU shall be rejected by the initiator sending an LMP_not_accepted PDU with the error code *pairing not allowed*. See Sequence 24.



Sequence 24: Both devices have a fixed PIN.

9.3.2.2.3 Responder rejects pairing

If the responder rejects pairing, it shall send an LMP_not_accepted PDU with the error code *pairing not allowed* after receiving an LMP_in_rand PDU. See Sequence 25.



Sequence 25: Responder rejects pairing.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.2.2.4 Creation of the link key

When K_{init} is calculated in both devices, the link key shall be created. This link key will be used in the authentication between the two devices for all subsequent connections until it is changed (see 9.3.2.3 and 9.3.2.4). The link key created in the pairing procedure will be either a combination key or one of the device's unit keys. The following rules shall apply to the selection of the link key:

— If one device sends an LMP_unit_key PDU and the other device sends LMP_comb_key PDU, the unit key will be the link key.
— If both devices send an LMP_unit_key PDU, the master's unit key will be the link key.
— If both devices send an LMP_comb_key PDU, the link key shall be calculated as described in 13.3.2.

The content of the LMP_unit_key PDU is the unit key bitwise XORed with K_{init}. The content of the LMP_comb_key PDU is LK_RAND bitwise XORed with K_{init}. Any device configured to use a combination key shall store the link key.

The use of unit keys is deprecated since it is implicitly insecure.

When the link key (i.e., combination or unit key) has been created, mutual authentication shall be performed to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication is performed with the initiator as the verifier. When finalized, an authentication in the reverse direction is performed. See Sequence 26.



Sequence 26: Creation of the link key.

9.3.2.2.5 Repeated attempts

When the authentication after creation of the link key fails because of an incorrect authentication response, the same scheme as in 9.3.2.1.3 shall be used. This prevents an intruder from trying a large number of different PINs in a relatively short time.

9.3.2.3 Change link key

If the link key is derived from combination keys and the current link key is the semi-permanent link key, the link key can be changed. If the link key is a unit key, the devices shall go through the pairing procedure in order to change the link key. The contents of the LMP_comb_key PDU is protected by a bitwise XOR with the current link key.

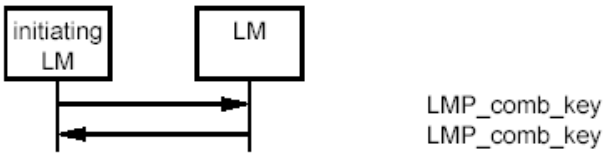
All sequences described in 9.3.2.3, including the mutual authentication after the link key has been changed, shall form a single transaction. The transaction ID from the first LMP_comb_key PDU shall be used for all subsequent sequences. See Table 48, Sequence 27, and Sequence 28.

Table 48—PDUs used to change link key

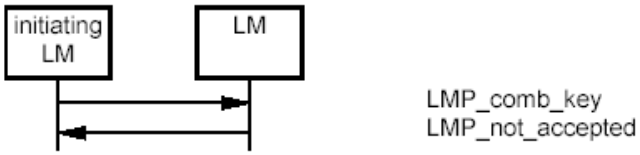
M/O	PDU	Contents
M	LMP_comb_key	Random number

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 27: Successful change of the link key.



Sequence 28: Change of the link key not possible since the other device uses a unit key.

If the change of link key is successful, the new link key shall be stored, and the old link key shall be discarded. The new link key shall be used as link key for all the following connections between the two devices until the link key is changed again. The new link key also becomes the current link key. It will remain the current link key until the link key is changed again or until a temporary link key is created (see 9.3.2.4).

When the new link key has been created, mutual authentication shall be performed to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication is performed with the device that initiated the link key change as verifier. When finalized, an authentication in the reverse direction is performed.

9.3.2.4 Change current link key type

The current link key can be a semi-permanent link key or a temporary link key. It may be changed temporarily, but the change shall be valid only for the current connection (see 13.3.1). Changing to a temporary link key is necessary if the piconet is to support encrypted broadcast. The current link key may not be changed before the connection establishment procedure has completed. This feature is supported only if broadcast encryption is supported as indicated by the LMP features mask. See Table 49.

Table 49—PDUs used to change current link key

M/O	PDU	Contents
O(23)	LMP_temp_rand	Random number
O(23)	LMP_temp_key	Key
O(23)	LMP_use_semi_permanent_key	—

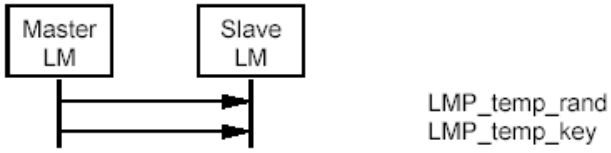
9.3.2.4.1 Change to a temporary link key

The master starts by creating the master key K_{master} as specified in Equation (23). Then the master shall generate a random number, RAND, and shall send it to the slave in an LMP_temp_rand PDU. Both sides then calculate an overlay denoted OVL as OVL = E_{22}(current link key, RAND, 16). The master shall then send K_{master} protected by a modulo-2 addition with OVL to the slave in an LMP_temp_key PDU. The slave

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

calculates K_{master}, based on OVL, which becomes the current link key. It shall be the current link key until the devices fall back to the semi-permanent link key (see 9.3.2.4.2). See Sequence 29.

NOTE—The terminology in this subclause is the same as used in 13.3.2.8.



Sequence 29: Change to a temporary link key.

All sequences described in 9.3.2.4.1, including the mutual authentication after K_{master} has been created, shall form a single transaction. The transaction ID shall be set to 0.

When the devices have changed to the temporary key, a mutual authentication shall be made to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication shall be performed with the master as verifier. When finalized, an authentication in the reverse direction is performed.

Should the mutual authentication fail at either side, the LM of the verifier should start the detach procedure. Because authentication is mutual, having the verifier initiate a detach will ensure the detach occurs if one of the devices is erroneous

9.3.2.4.2 Make the semi-permanent link key the current link key

After the current link key has been changed to K_{master}, this change can be undone, and the semi-permanent link key becomes the current link key again. If encryption is used on the link, the procedure to go back to the semi-permanent link key shall be immediately followed by the procedure where the master stops encryption (see 9.3.2.5.4). Encryption may be restarted by the master according to the procedures in 9.3.2.5.1. This is to assure that encryption with encryption parameters known by other devices in the piconet is not used when the semi-permanent link key is the current link key. See Sequence 30.



Sequence 30: Link key changed to the semi-permanent link key.

9.3.2.5 Encryption

If at least one authentication has been performed, encryption may be used. If the master intends to broadcast encrypted data, then it must use the same encryption parameters for all slaves in the piconet. In this case, it shall issue a temporary key, K_{master}, and shall make this key the current link key for all slaves in the piconet before encryption is started (see 9.3.2.4). See Table 50.

All sequences described in 9.3.2.5 shall form a single transaction. The transaction ID from the LMP_encryption_mode_req PDU shall be used for all subsequent sequences.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 50—PDUs used for handling encryption

M/O	PDU	Contents
O	LMP_encryption_mode_req	Encryption mode
O	LMP_encryption_key_size_req	Key size
O	LMP_start_encryption_req	Random number
O	LMP_stop_encryption_req	—

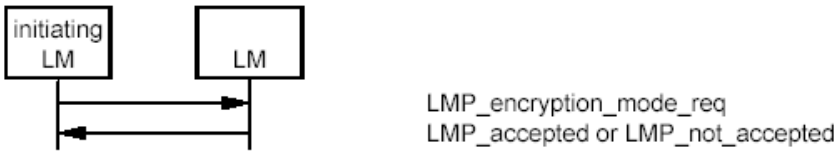
9.3.2.5.1 Encryption mode

The master and the slave must agree upon whether to use encryption (encryption mode = 1 in the LMP_encryption_mode_req PDU) or not (encryption mode = 0). If the semi-permanent key is used (Key_Flag = 0x00), encryption shall apply only to point-to-point packets. If the master link key is used (Key_Flag = 0x01), encryption shall apply to both point-to-point packets and broadcast packets. If master and slave agree on the encryption mode, the master continues to give more detailed information about the encryption.

Devices should never send an LMP_encryption_mode_req PDU with an encryption mode value of 2; however, for backwards compatibility, if the LMP_encryption_mode_req PDU is received with an encryption mode value of 2, then it should be treated the same as an encryption mode value of 1.

The initiating LM shall pause traffic on the ACL-U logical link (see 8.5.3.1). The initiating device shall then send the LMP_encryption_mode_req PDU. If the responding device accepts the change in encryption mode, then it shall complete the transmission of the current packet on the ACL logical transport and shall then suspend transmission on the ACL-U logical link. The responding device shall then send the LMP_accepted PDU.

ACL-U logical link traffic shall be resumed only after the attempt to encrypt or decrypt the logical transport is completed, i.e., at the end of Sequence 31, Sequence 32, or Sequence 33.



Sequence 31: Negotiation for encryption mode.

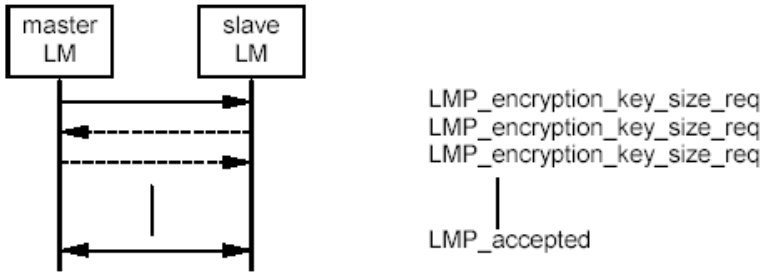
After a device has sent an LMP_encryption_mode_req PDU, it shall not send an LMP_au_rand PDU before encryption is started. After a device has received an LMP_encryption_mode_req PDU and sent an LMP_accepted PDU, it shall not send an LMP_au_rand PDU before encryption is started. If an LMP_au_rand PDU is sent violating these rules, the claimant shall respond with an LMP_not_accepted PDU with the error code *PDU not allowed*. This assures that devices will not have different ACOs when they calculate the encryption key. If the encryption mode is not accepted or the encryption key size negotiation results in disagreement, the devices may send an LMP_au_rand PDU again.

9.3.2.5.2 Encryption key size

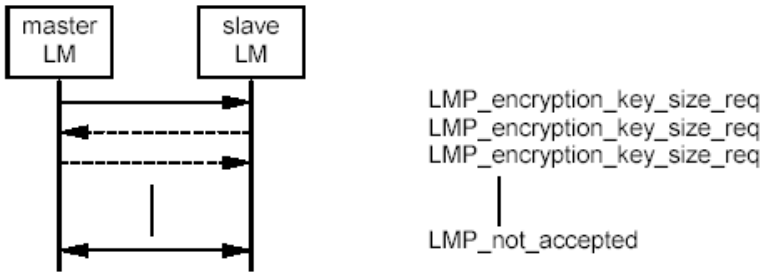
NOTE—This subclause uses the same terms as in 13.4.1.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The master sends an LMP_encryption_key_size_req PDU including the suggested key size $L_{sug,\ m}$, that is initially equal to $L_{max,\ m}$. If $L_{min,\ s} \leq L_{sug,\ m}$ and the slave supports $L_{sug,\ m}$, it shall respond with an LMP_accepted PDU, and $L_{sug,\ m}$ shall be used as the key size. If both conditions are not fulfilled, the slave sends back an LMP_encryption_key_size_req PDU including the slave's suggested key size $L_{sug,\ s}$. This value shall be the slave's largest supported key size that is less than $L_{sug,\ m}$. Then the master performs the corresponding test on the slave's suggestion. This procedure is repeated until a key size agreement is reached or it becomes clear that no such agreement can be reached. If an agreement is reached, a device sends an LMP_accepted PDU, and the key size in the last LMP_encryption_key_size_req PDU shall be used. After this, encryption is started (see 9.3.2.5.3). If an agreement is not reached, a device sends an LMP_not_accepted PDU with the error code *unsupported parameter value*, and the devices shall not communicate using encryption. See Sequence 32 and Sequence 33.



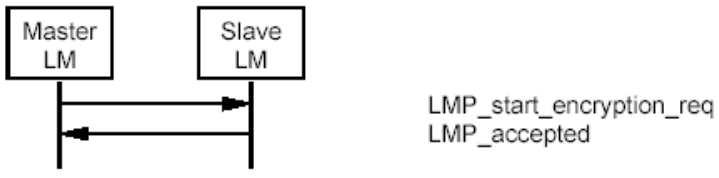
Sequence 32: Encryption key size negotiation successful.



Sequence 33: Encryption key size negotiation failed.

9.3.2.5.3 Start encryption

To start encryption, the master issues the random number EN_RAND and calculates the encryption key. See 13.3.2.5. The random number shall be the same for all slaves in the piconet when broadcast encryption is used. The master issues EN_RAND by sending an LMP_start_encryption_req PDU including EN_RAND. The slave shall calculate the encryption key when this message is received and shall acknowledge with an LMP_accepted PDU. See Sequence 34.



Sequence 34: Start of encryption.

Starting encryption shall be performed in three steps:

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

a) Master is configured to transmit unencrypted packets and to receive encrypted packets.
b) Slave is configured to transmit and receive encrypted packets.
c) Master is configured to transmit and receive encrypted packets.

Between step a and step b, master-to-slave transmission is possible. This is when an LMP_start_ encryption_req PDU is transmitted. Step b is triggered when the slave receives this message. Between step b and step c, slave-to-master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step c is triggered when the master receives this message.

9.3.2.5.4 Stop encryption

To stop encryption, a device shall send an LMP_encryption_mode_req PDU with the parameter encryption mode equal to 0 (no encryption). The other device responds with an LMP_accepted PDU or an LMP_not_accepted PDU (the procedure is described in Sequence 31 in 9.3.2.5.1). If accepted, encryption shall be stopped when the master sends an LMP_stop_encryption_req PDU, and the slave shall respond with an LMP_accepted PDU according to Sequence 35.



Sequence 35: Stop of encryption.

Stopping encryption shall be performed in three steps, similar to the procedure for starting encryption.

a) Master is configured to transmit encrypted packets and to receive unencrypted packets.
b) Slave is configured to transmit and receive unencrypted packets.
c) Master is configured to transmit and receive unencrypted packets.

Between step a and step b, master-to-slave transmission is possible. This is when an LMP_stop_ encryption_req PDU is transmitted. Step b is triggered when the slave receives this message. Between step b and step c, slave-to-master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step c is triggered when the master receives this message.

9.3.2.5.5 Change encryption mode, key, or random number

If the encryption key or encryption random number need to be changed or if the current link key needs to be changed according to the procedures in 9.3.2.4, encryption shall be stopped and restarted after completion, using the procedures in 9.3.2.5.3 and 9.3.2.5.4, for the new parameters to take effect.

NOTE—Because the ACL-C channel has priority over the ACL-U channel, it is possible for data to be queued up in the protocol stack at the point when encryption is stopped. Such data could then be sent in the clear between encryption being stopped and restarted.

When broadcast encryption is supported via the LMP features mask, it is possible for the master to request a slave's supported encryption key sizes. See Table 51.

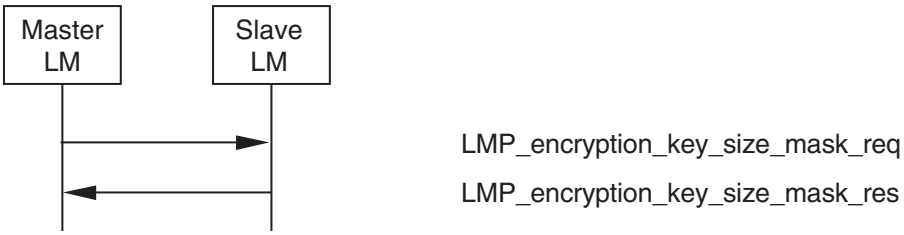
The master shall send an LMP_key_size_req PDU to the slave to obtain the slaves supported encryption key sizes.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 51—PDUs used for encryption key size request

M/O	PDU	Contents
O(23)	LMP_encryption_key_size_mask_req	
O(23)	LMP_encryption_key_size_mask_res	Key size mask

The slave shall return a bit mask indicating all broadcast encryption key sizes supported. The LSB shall indicate support for a key size of 1, the next MSB shall indicate support for a key size of 2 ,and so on up to a key size of 16. In all cases, a bit set to 1 shall indicate support for a key size; a bit set to 0 shall indicate that the key size is not supported. See Sequence 36.



Sequence 36: Request for supported encryption key sizes.

9.3.3 Informational requests

9.3.3.1 Timing accuracy

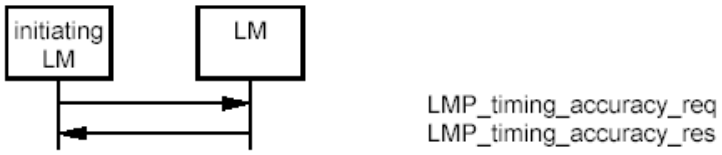
LMP supports requests for the timing accuracy. This information can be used to minimize the scan window during piconet physical channel resynchronization (see 8.2.2.5.2). The timing accuracy parameters returned are the long-term drift measured in parts per million and the long-term jitter measured in microseconds of the worst-case clock used. These parameters are fixed for a certain device and shall be identical when requested several times. Otherwise, the requesting device shall assume worst-case values (drift = 250 ppm and jitter = 10 μs). See Table 52, Sequence 37, and Sequence 38.

Table 52—PDUs used for requesting timing accuracy information

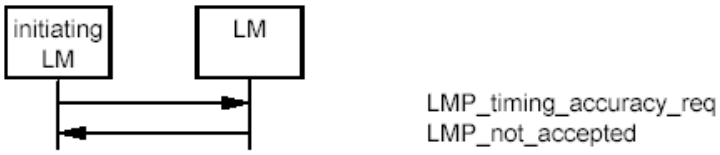
M/O	PDU	Contents
O(4)	LMP_timing_accuracy_req	—
O(4)	LMP_timing_accuracy_res	Drift Jitter

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 37: The requested device supports timing accuracy information.



Sequence 38: The requested device does not support timing accuracy information.

9.3.3.2 Clock offset

The clock offset can be used to speed up the paging time the next time the same device is paged. The master can request the clock offset at anytime following a successful BB paging procedure (i.e., before, during, or after connection setup). The clock offset shall be defined by the following equation:

$(CLKN_{16-2\ slave} - CLKN_{16-2\ master})$ mod 2**15.

See Table 53 and Sequence 39.

Table 53—PDUs used for clock offset request

M/O	PDU	Contents
M	LMP_clkoffset_req	—
M	LMP_clkoffset_res	Clock offset



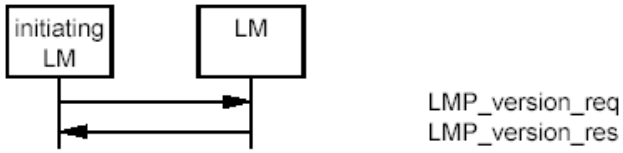
Sequence 39: Clock offset requested.

9.3.3.3 LMP version

LMP supports requests for the version of the LMP. The LMP_version_req and LMP_version_res PDUs contain three parameters: VersNr, CompId, and SubVersNr. VersNr specifies the version of the IEEE 802.15.1-2005 LMP specification that the device supports. CompId is used to track possible problems with the lower layers. All companies that create a unique implementation of the LM shall have their own

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

CompId. The same company is also responsible for the administration and maintenance of the SubVersNr. It is recommended that each company have a unique SubVersNr for each RF/BB/LM implementation. For a given VersNr and CompId, the values of the SubVersNr shall increase each time a new implementation is released. For both CompId and SubVersNr, the value 0xFFFF means that no valid number applies. There is no ability to negotiate the version of the LMP. Sequence 40 is used only to exchange the parameters. LMP version can be requested at any time following a successful BB paging procedure. See Table 54.



Sequence 40: Request for LMP version.

Table 54—PDUs used for LMP version request

M/O	PDU	Contents
M	LMP_version_req	VersNr CompId SubVersNr
M	LMP_version_res	VersNr CompId SubVersNr

9.3.3.4 Supported features

The supported features may be requested at any time following a successful BB paging procedure by sending the LMP_features_req PDU. Upon reception of an LMP_features_req PDU, the receiving device shall return an LMP_features_res PDU.

The number of features bits required may in the future exceed the size of a single page of features. An extended features mask is, therefore, provided to allow support for more than 64 features. Support for the extended features mask is indicated by the presence of the appropriate bit in the LMP features mask. The LMP_features_req_ext and LMP_features_res_ext PDUs operate in precisely the same way as the LMP_features_req and LMP_features_res PDUs except that they allow the various pages of the extended features mask to be requested. The LMP_features_req_ext PDU may be sent at any time following the exchange of the LMP_features_req and LMP_features_rsp PDUs.

The LMP_features_req_ext PDU contains a feature page index that specifies which page is requested and the contents of that page for the requesting device. Pages are numbered from 0–255 with page 0 corresponding to the normal features mask. Each page consists of 64 bits. If a device supports the LMP_features_res_ext PDU, but does not support any page number, it shall return a mask with every bit set to 0. If a device does not support the LMP_features_res_ext PDU, it shall return an LMP_not_accepted PDU. It also contains the maximum features page number containing any nonzero bit for this device. The recipient of an LMP_features_req_ext PDU shall respond with an LMP_features_res_ext PDU containing the same page number and the appropriate features page along with its own maximum features page number.

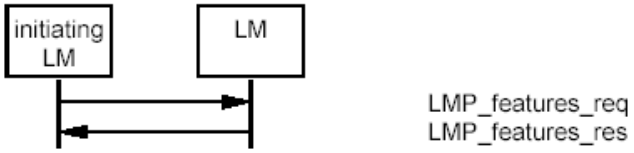
If the extended features request is not supported, then all bits in all extended features pages for that device shall be assumed to be zero. See Table 55, Sequence 41, and Sequence 42.

Copyright © 2005 IEEE. All rights reserved.

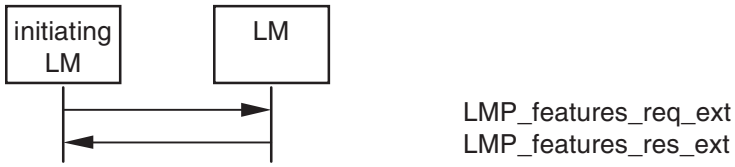
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 55—PDUs used for features request

M/O	PDU	Contents
M	LMP_features_req	Features
M	LMP_features_res	Features
O(63)	LMP_features_req_ext	Features page Maximum supported page Extended features
O(63)	LMP_features_res_ext	Features page Maximum supported page Extended features



Sequence 41: Request for supported features.



Sequence 42: Request for extended features.

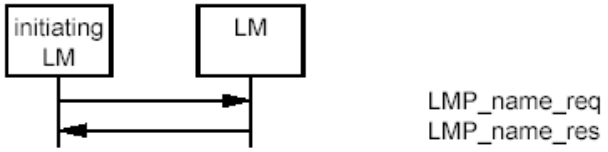
9.3.3.5 Name request

LMP supports name request to another device. The name is a user-friendly name associated with the device and consists of a maximum of 248 bytes coded according to the UTF-8 standard. The name is fragmented over one or more **DM1** packets. When an LMP_name_req PDU is sent, a name offset indicates which fragment is expected. The corresponding LMP_name_res PDU carries the same name offset, the name length indicating the total number of bytes in the name of the device, and the name fragment, where

— Name fragment(N) = name(N + name offset), if (N + name offset) < name length
— Name fragment(N) = 0, otherwise.

Here $0 \le N \le 13$. In the first sent LMP_name_req PDU, name offset = 0. Sequence 43 is then repeated until the initiator has collected all fragments of the name. The name request may be made at any time following a successful BB paging procedure. See Table 56.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 43: Device's name requested and responses.

Table 56—PDUs used for name request

M/O	PDU	Contents
M	LMP_name_req	Name offset
M	LMP_name_res	Name offset Name length Name fragment

9.3.4 Role switch

This subclause describes the LMP sequences used for role switch including the slot offset command used to retrieve timing information and the role switch commands used to reverse roles.

9.3.4.1 Slot offset

With the LMP_slot_offset PDU, the information about the difference between the slot boundaries in different piconets is transmitted. The LMP_slot_offset PDU may be sent anytime after the BB paging procedure has completed. This PDU carries the parameters slot offset and BD_ADDR. The slot offset shall be the time in microseconds between the start of a master transmission in the current piconet to the start of the next following master transmission in the piconet where the BD_ADDR device (normally the slave) is master at the time that the request is interpreted by the BD_ADDR device. See Figure 90.

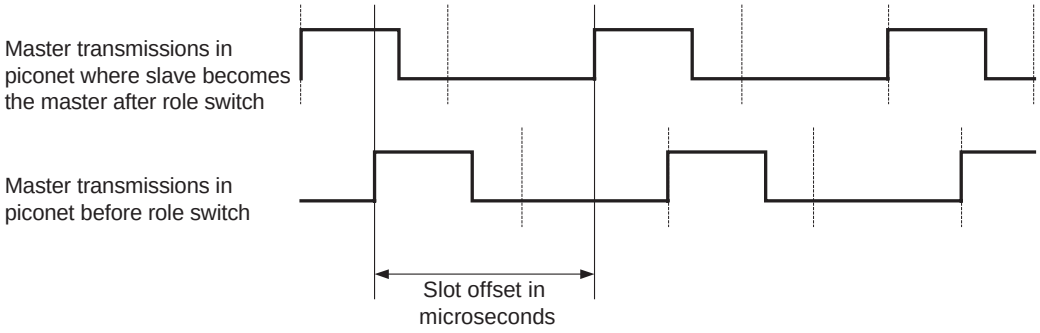


Figure 90—Slot offset for role switch

See 9.3.4 for the use of the LMP_slot_offset PDU in the context of the role switch. In the case of role switch, the BD_ADDR is that of the slave device. See Table 57 and Sequence 44.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 57— PDU used for slot offset information

M/O	PDU	Contents
O(3)	LMP_slot_offset	Slot offset BD_ADDR



Sequence 44: Slot offset information is sent.

9.3.4.2 Role switch

Since the paging device always becomes the master of the piconet, a switch of the master-slave role is sometimes needed (see 8.8.6.5). The LMP_switch_req PDU may be sent anytime after the BB paging procedure has completed.

Support for the LMP_slot_offset PDU is mandatory if the LMP_switch_req PDU is supported.

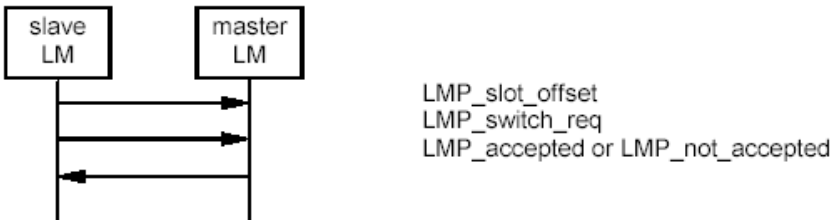
The LMP_slot_offset PDU shall be sent only if the ACL logical transport is in active mode. The LMP_switch_req PDU shall be sent only if the ACL logical transport is in active mode, when encryption is disabled, and all synchronous logical transports on the same physical link are disabled. Additionally, the LMP_slot_offset or LMP_switch_req PDU shall not be initiated or accepted while a synchronous logical transport is being negotiated by the LM. See Table 58.

Table 58—PDUs used for role switch

M/O	PDU	Contents
O(5)	LMP_switch_req	Switch instant
O(5)	LMP_slot_offset	Slot offset BD_ADDR

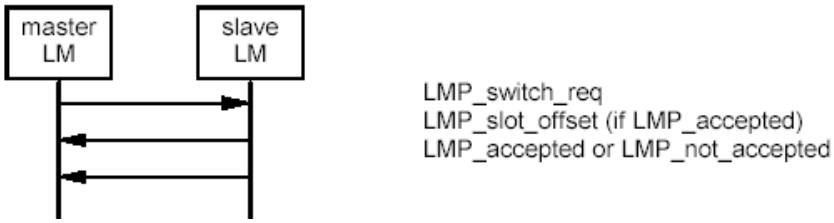
The initiating LM shall pause traffic on the ACL-U logical link (see 8.5.3.1). It shall then send an LMP_slot_offset PDU immediately followed by an LMP_switch_req PDU. If the master accepts the role switch, it shall pause traffic on the ACL-U logical link (see 8.5.3.1) and respond with an LMP_accepted PDU. When the role switch has been completed at the BB level (successfully or not), both devices reenable transmission on the ACL-U logical link. If the master rejects the role switch, it responds with an LMP_not_accepted PDU, and the slave reenables transmission on the ACL-U logical link. The transaction ID for all PDUs in the sequence shall be set to 1. See Sequence 45.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 45: Role switch (slave initiated).

If the master initiates the role switch, it shall pause traffic on the ACL-U logical link (see 8.5.3.1) and send an LMP_switch_req PDU. If the slave accepts the role switch, it shall pause traffic on the ACL-U logical link (see 8.5.3.1) and responds with an LMP_slot_offset PDU immediately followed by an LMP_accepted PDU. When the role switch has been completed at the BB (successfully or not), both devices reenable transmission on the ACL-U logical link. If the slave rejects the role switch, it responds with an LMP_not_accepted PDU, and the master reenables transmission on the ACL-U logical link. The transaction ID for all PDUs in the sequence shall be set to 0. See Sequence 46.



Sequence 46: Role switch (master initiated).

The LMP_switch_req PDU contains a parameter, switch instant, that specifies the instant at which the TDD switch is performed. This is specified as CLK, which is available to both devices. This instant is chosen by the sender of the message and shall be at least $2*T_{poll}$ or 32 (whichever is greater) slots in the future. The switch instant shall be within 12 hr of the current clock value to avoid clock wrap.

The sender of the LMP_switch_req PDU selects the switch instant, queues the LMP_switch_req PDU to link control for transmission, and starts a timer to expire at the switch instant. When the timer expires, it initiates the mode switch. In the case of a master-initiated switch, if the LMP_slot_offset PDU has not been received by the switch instant, the role switch is carried out without an estimate of the slave's slot offset. If an LMP_not_accepted PDU is received before the timer expires, then the timer is stopped, and the role switch shall not be initiated.

When the LMP_switch_req is received, the switch instant is compared with the CLK. If it is in the past, then the instant has been passed, and an LMP_not_accepted PDU with the error code *instant passed* shall be returned. If it is in the future and the role switch is allowed, then an LMP_accepted PDU shall be returned, and a timer is started to expire at the switch instant. When this timer expires, the role switch shall be initiated.

After a successful role switch, the supervision timeout and poll interval T_{poll} shall be set to their default values. The authentication state and the ACO shall remain unchanged. AFH shall follow the procedures described in 8.8.6.5. The default value for the max_slots parameter shall be used.

9.3.5 Modes of operation

This subclause describes the LMP sequences required to put an active link into HOLD mode, PARK state, or SNIFF mode, along with the LMP sequences used to communicate with slaves in PARK state.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.5.1 HOLD mode

The ACL logical transport of a connection between two devices can be placed in HOLD mode for a specified hold time. See 8.8.8 for details. See also Table 59.

Table 59—PDUs used for HOLD mode

M/O	PDU	Contents
O(6)	LMP_hold	Hold time, hold instant
O(6)	LMP_hold_req	Hold time, hold instant

The LMP_hold and LMP_hold_req PDUs both contain a parameter, hold instant, that specifies the instant at which the HOLD mode becomes effective. This is specified as CLK, which is available to both devices. The hold instant is chosen by the sender of the message and should be at least $6*T_{poll}$ slots in the future. The hold instant shall be within 12 hr of the current clock value to avoid clock wrap.

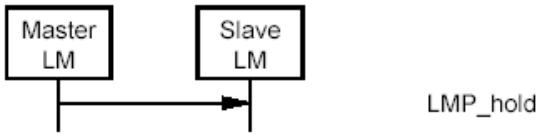
9.3.5.1.1 Master forces HOLD mode

The master may force HOLD mode if there has previously been a request for HOLD mode that has been accepted. The hold time included in the PDU when the master forces HOLD mode shall not be longer than any hold time the slave has previously accepted when there was a request for HOLD mode. See Sequence 47.

The master LM shall first pause traffic on the ACL-U logical link (see 8.5.3.1). It shall select the hold instant and queue the LMP_hold PDU to its link control for transmission. It shall then start a timer to wait until the hold instant occurs. When this timer expires, then the connection shall enter HOLD mode. If the BB acknowledgment for the LMP_hold PDU is not received, then the master may enter HOLD mode, but it shall not use its low accuracy clock during the HOLD mode.

When the slave LM receives an LMP_hold PDU, it compares the hold instant with the current CLK value. If it is in the future, then it starts a timer to expire at this instant and enters HOLD mode when it expires.

When the master LM exits from HOLD mode, it reenables transmission on the ACL-U logical link.



Sequence 47: Master forces slave into HOLD mode.

9.3.5.1.2 Slave forces HOLD mode

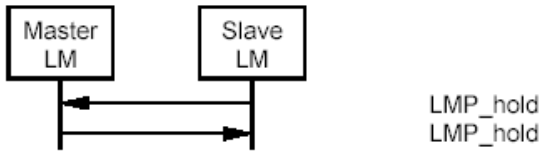
The slave may force HOLD mode if there has previously been a request for HOLD mode that has been accepted. The hold time included in the PDU when the slave forces HOLD mode shall not be longer than any hold time the master has previously accepted when there was a request for HOLD mode. See Sequence 48.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The slave LM shall first complete the transmission of the current packet on the ACL logical transport and then shall suspend transmission on the ACL-U logical link. It shall select the hold instant and queue the LMP_hold PDU to its link control for transmission. It shall then wait for an LMP_hold PDU from the master acting according to the procedure described in 9.3.5.1.1.

When the master LM receives an LMP_hold PDU, it shall pause traffic on the ACL-U logical link (see 8.5.3.1). It shall then inspect the hold instant. If this is less than $6*T_{poll}$ slots in the future, it shall modify the instant so that it is at least $6*T_{poll}$ slots in the future. It shall then send an LMP_hold PDU using the mechanism described in 9.3.5.1.1.

When the master and slave LMs exit from HOLD mode, they shall reenable transmission on the ACL-U logical link.



Sequence 48: Slave forces master into HOLD mode.

9.3.5.1.3 Master or slave requests HOLD mode

The master or the slave can request to enter HOLD mode. Upon receipt of the request, the same request with modified parameters can be returned, or the negotiation can be terminated. If an agreement is seen, an LMP_accepted PDU terminates the negotiation, and the ACL link is placed in HOLD mode. If no agreement is seen, an LMP_not_accepted PDU with the error code *unsupported parameter value* terminates the negotiation, and HOLD mode is not entered.

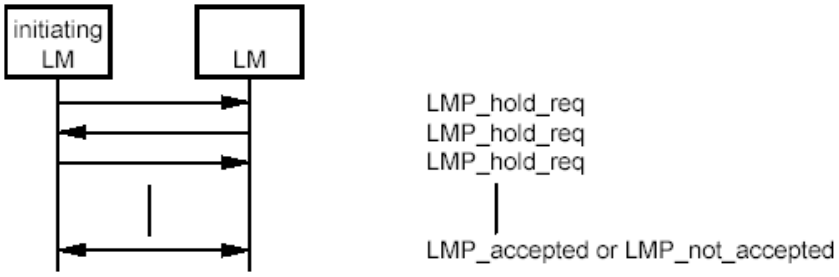
The initiating LM shall pause traffic on the ACL-U logical link (see 8.5.3.1). On receiving an LMP_hold_req PDU, the receiving LM shall complete the transmission of the current packet on the ACL logical transport and then shall suspend transmission on the ACL-U logical link.

The LM sending the LMP_hold_req PDU selects the hold instant that shall be at least $9*T_{poll}$ slots in the future. If this is a response to a previous LMP_hold_req PDU and the hold instant contained is at least $9*T_{poll}$ slots in the future, then this shall be used. LMP_hold_req PDU shall then be queued to its link control for transmission, and it shall start a timer to expire at the hold instant. When the timer expires, the connection shall enter HOLD mode unless an LMP_not_accepted or LMP_hold_req PDU is received before expiry. If the LM receiving LMP_hold_req PDU agrees to enter HOLD mode, it shall return an LMP_accepted PDU and shall start a timer to expire at the hold instant. When this timer expires, it enters HOLD mode.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When each LM exits from HOLD mode, it shall reenable transmission on the ACL-U logical link. See Sequence 49.



Sequence 49: Negotiation for HOLD mode.

9.3.5.2 PARK state

If a slave does not need to participate in the channel, but should still remain synchronized to the master, it may be placed in PARK state. See 8.8.9 for details.

To keep a parked slave connected, the master shall periodically unpark and repark the slave if the supervision timeout is not set to zero (see 8.3.1).

All PDUs sent from the master to parked slaves are carried on the control logical channel using the PSB logical transport (LMP link of PSB logical transport). These PDUs (i.e., LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_addr_req and LMP_unpark_ PM_addr_req) are the only PDUs that shall be sent to a slave in PARK state and the only LMP PDUs that shall be broadcast. To increase reliability for broadcast, the packets are as short as possible. Therefore, the format for these LMP PDUs are somewhat different. The parameters are not always byte-aligned, and the length of the PDUs is variable.

The messages for controlling PARK state include parameters, defined in 8.8.9. When a slave is placed in PARK state, it is assigned a unique PM_ADDR, which can be used by the master to unpark that slave. The all-zero PM_ADDR has a special meaning; it is not a valid PM_ADDR. If a device is assigned this PM_ADDR, it shall be identified with its BD_ADDR when it is unparked by the master. See Table 60.

Table 60—PDUs used for PARK state

M/O	PDU	Contents
O(8)	LMP_park_req	Timing control flags D_B T_B N_B Δ_B PM_ADDR AR_ADDR N_{Bsleep} D_{Bsleep} D_{access} T_{access} $N_{acc\text{-}slots}$ N_{poll} M_{access} Access scheme

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 60—PDUs used for PARK state *(continued)*

M/O	PDU	Contents
O(8)	LMP_set_broadcast_scan_window	Timing control flags D_B (optional) Broadcast scan window
O(8)	LMP_modify_beacon	Timing control flags D_B (optional) T_B N_B Δ_B D_{access} T_{access} $N_{acc\text{-}slots}$ N_{poll} M_{access} access scheme
O(8)	LMP_unpark_PM_ADDR_req	Timing control flags D_B (optional) LT_ADDR 1st unpark LT_ADDR 2nd unpark(optional) PM_ADDR 1st unpark PM_ADDR 2nd unpark(optional) LT_ADDR 3rd unpark(optional) LT_ADDR 4th unpark(optional) PM_ADDR 3rd unpark(optional) PM_ADDR 4th unpark(optional) LT_ADDR 5th unpark(optional) LT_ADDR 6th unpark(optional) PM_ADDR 5th unpark(optional) PM_ADDR 6th unpark(optional) LT_ADDR 7th unpark(optional) PM_ADDR 7th unpark(optional)
O(8)	LMP_unpark_BD_ADDR _req	Timing control flags D_B (optional) LT_ADDR LT_ADDR (optional) BD_ADDR BD_ADDR (optional)

9.3.5.2.1 Master requests slave to enter PARK state

The master can request PARK state. The master LM shall pause traffic on the ACL-U logical link (see 8.5.3.1) and then send an LMP_park_req PDU. If the slave agrees to enter PARK state, it shall pause traffic on the ACL-U logical link (see 8.5.3.1) and then respond with an LMP_accepted PDU.

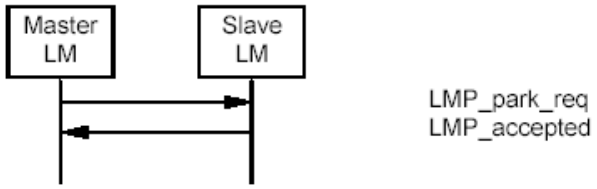
When the slave queues an LMP_accepted PDU, it shall start a timer for $6*T_{poll}$ slots. If the BB acknowledgment is received before this timer expires, the slave shall enter PARK state immediately; otherwise, it shall enter PARK state when the timer expires.

When the master receives an LMP_accepted PDU, it shall start a timer for $6*T_{poll}$ slots. When this timer expires, the slave is in PARK state, and the LT_ADDR may be reused.

Copyright © 2005 IEEE. All rights reserved.

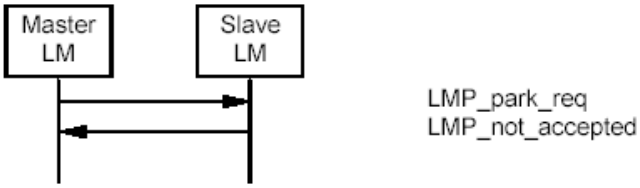
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

If the master never receives an LMP_accepted PDU, then a link supervision timeout will occur. See Sequence 50.



Sequence 50: Slave accepts to enter PARK state.

If the slave rejects the attempt to enter PARK state, it shall respond with an LMP_not_accepted PDU, and the master shall reenable transmission on the ACL-U logical link. See Sequence 51.



Sequence 51: Slave rejects to enter into PARK state.

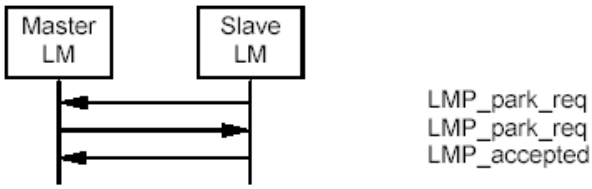
9.3.5.2.2 Slave requests to enter PARK state

The slave can request PARK state. The slave LM shall pause traffic on the ACL-U logical link (see 8.5.3.1) and then send an LMP_park_req PDU. When sent by the slave, the parameters PM_ADDR and AR_ADDR are not valid and the other parameters represent suggested values. If the master accepts the slave's request to enter PARK state, it shall pause traffic on the ACL-U logical link (see 8.5.3.1) and then send an LMP_park_req PDU, where the parameter values may be different from the values in the PDU sent from the slave. If the slave can accept these parameters, it shall respond with an LMP_accepted PDU.

When the slave queues an LMP_accepted PDU for transmission, it shall start a timer for $6*T_{poll}$ slots. If the BB acknowledgment is received before this timer expires, it shall enter PARK state immediately; otherwise, it shall enter PARK state when the timer expires.

When the master receives an LMP_accepted PDU, it shall start a timer for $6*T_{poll}$ slots. When this timer expires, the slave is in PARK state, and the LT_ADDR may be reused.

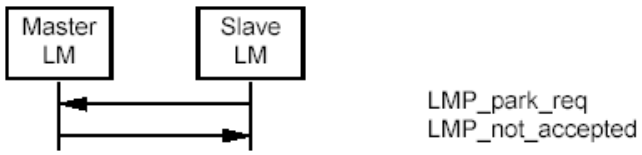
If the master never receives the LMP_accepted PDU, then a link supervision timeout will occur. See Sequence 52.



Sequence 52: Slave requests to enter PARK state and accepts master's beacon parameters.

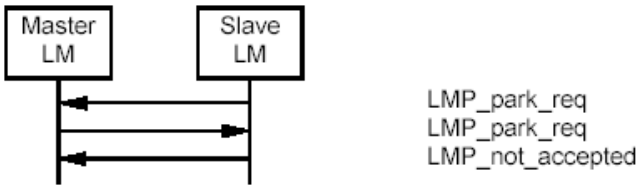
Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

If the master does not accept the slave's request to enter PARK state, it shall send an LMP_not_accepted PDU. The slave shall then reenable transmission on the ACL-U logical link. See Sequence 53.



Sequence 53: Master rejects slave's request to enter PARK state.

If the slave does not accept the parameters in the LMP_park_req PDU sent from the master, it shall respond with an LMP_not_accepted PDU, and both devices shall reenable transmission on the ACL-U logical link. See Sequence 54.



Sequence 54: Slave requests to enter PARK state, but rejects master's beacon parameters.

9.3.5.2.3 Master sets up broadcast scan window

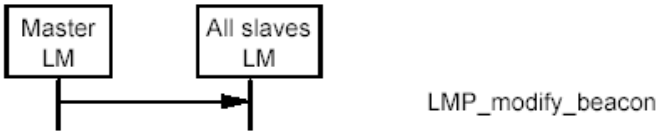
If more broadcast capacity is needed than the beacon train, the master may indicate to the slaves that more broadcast information will follow the beacon train by sending an LMP_set_broadcast_scan_window PDU. This message shall be sent in a broadcast packet at the beacon slot(s). The scan window shall start in the beacon instant and shall be valid only for the current beacon. See Sequence 55.



Sequence 55: Master notifies all slaves of increase in broadcast capacity.

9.3.5.2.4 Master modifies beacon parameters

When the beacon parameters change, the master notifies the parked slaves of this by sending an LMP_modify_beacon PDU. This PDU shall be sent in a broadcast packet. See Sequence 56.



Sequence 56: Master modifies beacon parameters.

Copyright © 2005 IEEE. All rights reserved.

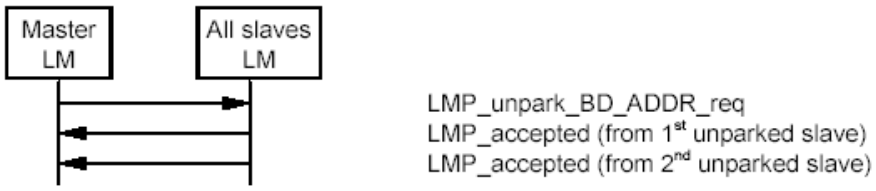
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.5.2.5 Unparking slaves

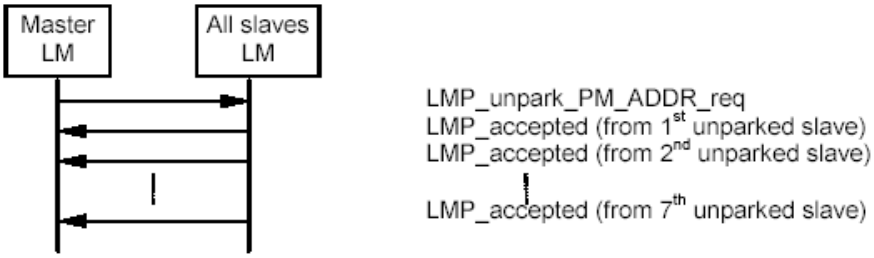
The master can unpark one or many slaves by sending a broadcast LMP message including the PM_ADDR or the BD_ADDR of the device(s) to be unparked. Broadcast LMP messages are carried on the control logical channel using the PSB logical transport. See 8.8.9.5 for further details. This message also includes the LT_ADDR that the master assigns to the slave(s). After sending this message, the master shall check the success of the unpark by polling each unparked slave by sending POLL packets, so that the slave is granted access to the channel. The unparked slave shall then send a response with an LMP_accepted PDU. If this message is not received from the slave within a *newconnectionTO* after the master sent the unpark message, the unpark failed, and the master shall consider the slave as still being in PARK state.

One PDU is used where the parked device is identified with the PM_ADDR, and another PDU is used where it is identified with the BD_ADDR. Both messages have variable length depending on the number of slaves the master unparks. For each slave the master wishes to unpark, an LT_ADDR, followed by the PM_ADDR or BD_ADDR of the device that is assigned this LT_ADDR, is included in the payload. If the slaves are identified with the PM_ADDR, a maximum of seven slaves can be unparked with the same message. If they are identified with the BD_ADDR, a maximum of two slaves can be unparked with the same message.

After a successful unparking, both master and slave reenable transmission on the ACL-U logical link. See Sequence 57 and Sequence 58.



Sequence 57: Master unparks slaves addressed with their BD_ADDR.



Sequence 58: Master unparks slaves addressed with their PM_ADDR.

9.3.5.3 SNIFF mode

To enter SNIFF mode, master and slave negotiate a sniff interval, T_{sniff}, and a sniff offset, D_{sniff}, that specifies the timing of the sniff slots. The offset determines the time of the first sniff slot; after that, the sniff slots follow periodically with the sniff interval T_{sniff}. To avoid clock wraparound during the initialization, one of two options is chosen for the calculation of the first sniff slot. A timing control flag in the message from the master indicates this. Only bit 1 of the timing control flag is valid.

When the ACL logical transport is in SNIFF mode, the master shall start a transmission only in the sniff slots. Two parameters control the listening activity in the slave: the sniff attempt and the sniff timeout. The sniff attempt parameter determines for how many slots the slave shall listen when the slave is not treating this as a scatternet link, beginning at the sniff slot, even if it does not receive a packet with its own LT_ADDR. The sniff timeout parameter determines for how many additional slots the slave shall listen

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

when the slave is not treating this as a scatternet link if it continues to receive only packets with its own LT_ADDR. It is not possible to modify the sniff parameters while the device is in SNIFF mode. See Table 61. (See 8.8.7 for more details of SNIFF mode behavior.)

Table 61—PDUs used for SNIFF mode

M/O	PDU	Contents
O(7)	LMP_sniff_req	Timing control flags D_{sniff} T_{sniff} Sniff attempt Sniff timeout
O(7)	LMP_unsniff_req	—

9.3.5.3.1 Master or slave requests SNIFF mode

Either the master or the slave may request entry to SNIFF mode.

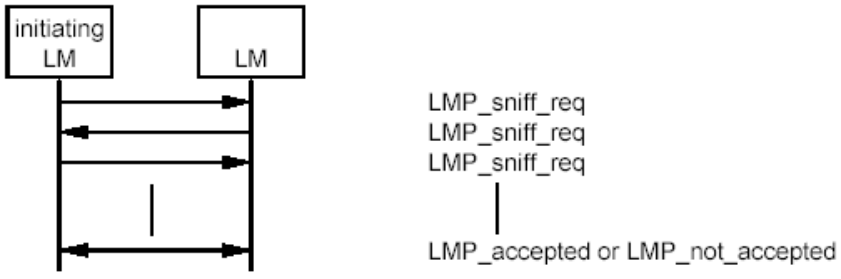
The process is initiated by sending an LMP_sniff_req PDU containing a set of parameters. The receiving LM shall then decide whether to reject the attempt by sending an LMP_not_accepted PDU, to suggest different parameters by replying with an LMP_sniff_req PDU, or to accept the request.

Before the first time that the master sends an LMP_sniff_req PDU, it shall enter SNIFF TRANSITION mode. If the master receives or sends an LMP_not_accepted PDU, it shall exit from SNIFF TRANSITION mode. If the master receives an LMP_sniff_req PDU, it shall enter SNIFF TRANSITION mode.

If the master decides to accept the request, it shall send an LMP_accepted PDU. When the master receives the BB acknowledgment for this PDU, it shall exit SNIFF TRANSITION mode and enter SNIFF mode.

If the master receives an LMP_accepted PDU, the master shall exit from SNIFF TRANSITION mode and enter SNIFF mode.

If the slave receives an LMP_sniff_req PDU, it must decide whether to accept the request. If the slave rejects SNIFF mode, then it replies with an LMP_not_accepted PDU. If the slave accepts SNIFF mode, but requires a different set of parameters, it shall respond with an LMP_sniff_req PDU containing the new parameters. If the slave decides that the parameters are acceptable, then it shall send an LMP_accepted PDU and enter SNIFF mode. If the slave receives an LMP_not_accepted PDU, it shall terminate the attempt to enter SNIFF mode. See Sequence 59.



Sequence 59: Negotiation for SNIFF mode.

Copyright © 2005 IEEE. All rights reserved.

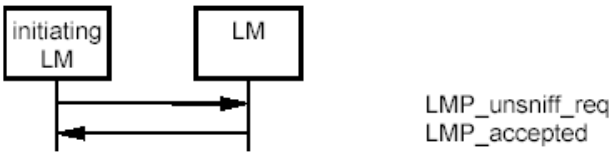
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.5.3.2 Moving a slave from SNIFF mode to active mode

SNIFF mode may be exited by either the master or the slave sending an LMP_unsniff_req PDU. The requested device must reply with an LMP_accepted PDU.

If the master requests an exit from SNIFF mode, it shall enter SNIFF TRANSITION mode and then send an LMP_unsniff_req PDU. When the slave receives the LMP_unsniff_req, it shall exit from SNIFF mode and reply with an LMP_accepted PDU. When the master receives the LMP_accepted PDU, it shall exit from SNIFF TRANSITION mode and enter active mode.

If the slave requests an exit from SNIFF mode, it shall send an LMP_unsniff_req PDU. When the master receives the LMP_unsniff_req PDU, it shall enter SNIFF TRANSITION mode and then send an LMP_accepted PDU. When the slave receives the LMP_accepted PDU, it shall exit from SNIFF mode and enter active mode. When the master receives the BB acknowledgment for the LMP_accepted PDU, it shall leave SNIFF TRANSITION mode and enter active mode. See Sequence 60.



Sequence 60: Slave moved from SNIFF mode to active mode.

9.3.6 Logical transports

When a connection is first established between two devices, the connection consists of the default ACL logical transport carrying the control logical link (for LMP messages) and the user logical link (for L2CAP data). One or more synchronous logical transports (SCO or eSCO) may then be added. A new logical transport shall not be created if it would cause all slots to be allocated to reserved slots on secondary LT_ADDRs.

9.3.6.1 SCO logical transport

The SCO logical transport reserves slots separated by the SCO interval T_{sco}. The first slot reserved for the SCO logical transport is defined by T_{sco} and the SCO offset D_{sco}. See 8.8.6.2 for details. A device shall initiate a request for **HV2** or **HV3** packet type only if the other device supports it (bit 12, bit 13) in its features mask. A device shall initiate CVSD, μ-law, or A-law coding or uncoded (transparent) data only if the other device supports the corresponding feature. To avoid problems with a wraparound of the clock during initialization of the SCO logical transport, the timing control flags parameter is used to indicate how the first SCO slot shall be calculated. Only bit 1 of the timing control flags parameter is valid. The SCO link is distinguished from all other SCO links by an SCO handle. The SCO handle zero shall not be used. See Table 62.

Table 62—PDUs used for managing the SCO links

M/O	PDU	Contents
O(11)	LMP_SCO_link_req	SCO handle Timing control flags D_{sco} T_{sco} SCO packet Air mode
O(11)	LMP_remove_SCO_link_req	SCO handle Error

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.6.1.1 Master initiates an SCO link

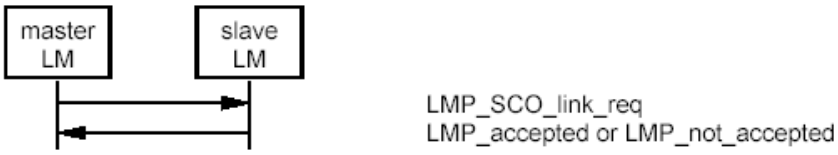
When establishing an SCO link, the master sends a request, a LMP_SCO_link_req PDU, with parameters that specify the timing, packet type, and coding that will be used on the SCO link. Each of the SCO packet types supports three different voice coding formats on the air-interface: μ-law log PCM, A-law log PCM, and CVSD. The air coding by log PCM or CVSD may be deactivated to achieve a transparent synchronous data link at 64 kb/s.

The slots used for the SCO links are determined by three parameters controlled by the master: T_{sco}, D_{sco}, and a flag indicating how the first SCO slot is calculated. After the first slot, the SCO slots follow periodically at an interval of T_{sco}.

If the slave does not accept the SCO link, but is willing to consider another possible set of SCO parameters, it can indicate what it does not accept in the error code field of LMP_not_accepted PDU. The master may then issue a new request with modified parameters.

The SCO handle in the message shall be different from existing SCO link(s).

If the SCO packet type is **HV1**, the LMP_accepted PDU shall be sent using the **DM1** packet. See Sequence 61.

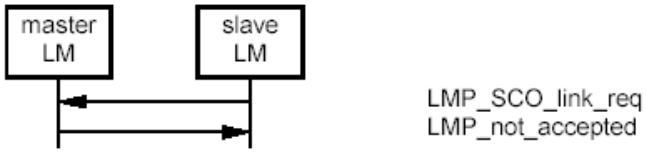


Sequence 61: Master requests an SCO link.

9.3.6.1.2 Slave initiates an SCO link

The slave may initiate the establishment of an SCO link. The slave sends an LMP_SCO_link_req PDU, but the parameters timing control flags and D_{sco} are invalid as well as the SCO handle, which shall be zero. If the master is not capable of establishing an SCO link, it replies with an LMP_not_accepted PDU. Otherwise, it sends back an LMP_SCO_link_req PDU. This message includes the assigned SCO handle, D_{sco}, and the timing control flags. The master should try to use the same parameters as in the slave request; if the master cannot meet that request, it is allowed to use other values. The slave shall then reply with LMP_accepted or LMP_not_accepted PDU.

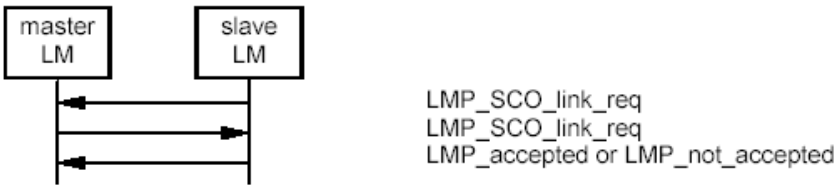
If the SCO packet type is **HV1**, the LMP_accepted shall be sent using the **DM1** packet. See Sequence 62 and Sequence 63.



Sequence 62: Master rejects slave's request for an SCO link.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.



Sequence 63: Master accepts slave's request for an SCO link.

9.3.6.1.3 Master requests change of SCO parameters

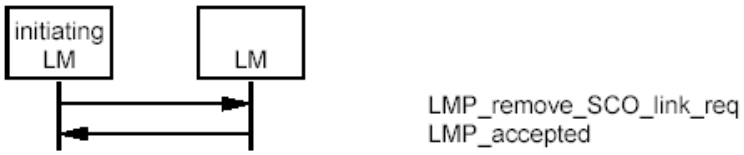
The master sends an LMP_SCO_link_req PDU where the SCO handle is the handle of the SCO link for which the master wishes to change parameters. If the slave accepts the new parameters, it replies with an LMP_accepted PDU, and the SCO link will change to the new parameters. If the slave does not accept the new parameters, it shall reply with an LMP_not_accepted PDU, and the SCO link is left unchanged. When the slave replies with an LMP_not_accepted PDU, it shall indicate in the error code parameter what it does not accept. The master may then try to change the SCO link again with modified parameters. The sequence is the same as in 9.3.6.1.1.

9.3.6.1.4 Slave requests change of SCO parameters

The slave sends an LMP_SCO_link_req PDU where the SCO handle is the handle of the SCO link to be changed. The parameters timing control flags and D_{sco} are not valid in this PDU. If the master does not accept the new parameters, it shall reply with an LMP_not_accepted PDU, and the SCO link is left unchanged. If the master accepts the new parameters, it shall reply with an LMP_SCO_link_req PDU containing the same parameters as in the slave request. When receiving this message, the slave replies with an LMP_not_accepted PDU if it does not accept the new parameters. The SCO link is then left unchanged. If the slave accepts the new parameters, it replies with an LMP_accepted PDU, and the SCO link will change to the new parameters. The sequence is the same as in 9.3.6.1.2.

9.3.6.1.5 Remove an SCO link

Master or slave can remove the SCO link by sending a request including the SCO handle of the SCO link to be removed and an error code indicating why the SCO link is removed. The receiving side shall respond with an LMP_accepted PDU. See Sequence 64.



Sequence 64: SCO link removed.

9.3.6.2 eSCO logical transport

After an ACL link has been established, one or more eSCO links can be set up to the remote device. The eSCO links are similar to SCO links using timing control flags, an interval T_{eSCO}, and an offset D_{eSCO}. Only bit 1 of the timing control flags parameter is valid. As opposed to SCO links, eSCO links have a configurable data rate that may be asymmetric and can be set up to provide limited retransmissions of lost or damaged packets inside a retransmission window of size W_{eSCO}. The D_{eSCO} shall be based on CLK. See Table 63.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

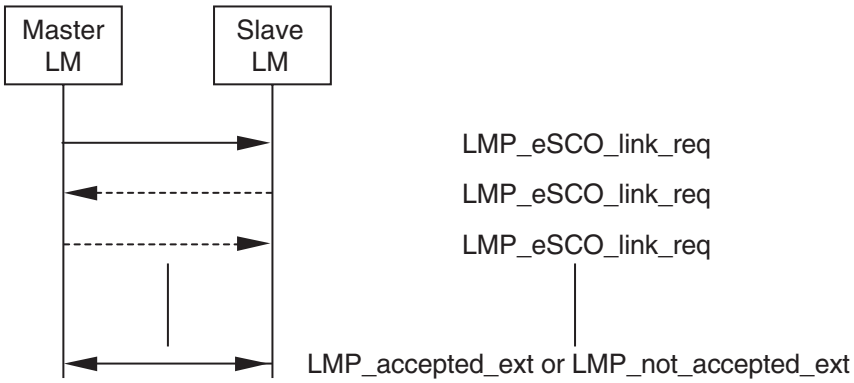
Table 63—PDUs used for managing the eSCO links

M/O	PDU	Contents
O(31)	LMP_eSCO_link_req	eSCO handle eSCO LT_ADDR Timing control flags D_{eSCO} T_{eSCO} W_{eSCO} eSCO packet type M→S eSCO packet type S→M Packet length M→S Packet length S→M Air mode Negotiation state
O(31)	LMP_remove_eSCO_link_req	eSCO handle Error

The parameters D_{eSCO}, T_{eSCO}, W_{eSCO}, eSCO packet type M→S, eSCO packet type S→M, packet length M→S, packet length S→M are henceforth referred to as the *negotiable parameters*.

9.3.6.2.1 Master initiates an eSCO link

When establishing an eSCO link, the master sends an LMP_eSCO_link_req PDU specifying all parameters. The slave may accept this with an LMP_accepted_ext PDU, reject it with an LMP_not_accepted_ext PDU, or respond with its own LMP_eSCO_link_req PDU specifying alternatives for some or all parameters. The slave shall not negotiate the eSCO handle or eSCO LT_ADDR parameters. The negotiation of parameters continues until the master or slave either accepts the latest parameters with an LMP_accepted_ext PDU or terminates the negotiation with an LMP_not_accepted_ext PDU. The negotiation shall use the procedures defined in 9.3.6.2.5. See Sequence 65.



Sequence 65: Master requests an eSCO link.

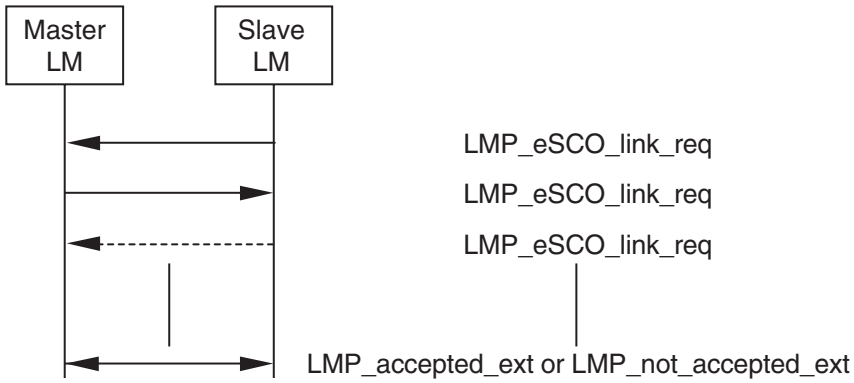
9.3.6.2.2 Slave initiates an eSCO link

When attempting to establish an eSCO link, the slave shall send an LMP_eSCO_link_req PDU specifying all parameters, with the exception of eSCO LT_ADDR and eSCO handle, which are invalid. The latter shall be set to zero. The master may respond to this with an LMP_eSCO_link_req PDU, filling in these missing parameters and potentially changing the other requested parameters. The slave may accept this with an

Copyright © 2005 IEEE. All rights reserved.

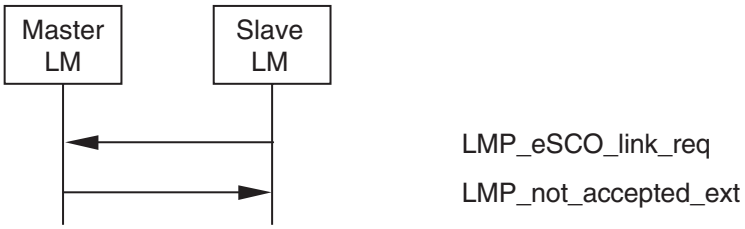
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

LMP_accepted_ext PDU or respond with a further LMP_eSCO_link_req PDU specifying alternatives for some or all of the parameters. The negotiation of parameters continues until the master or slave either accepts the latest parameters with an LMP_accepted_ext PDU or terminates the negotiation with an LMP_not_accepted_ext PDU. See Sequence 66.



Sequence 66: Slave requests an eSCO link.

The master may reject the request immediately with an LMP_not_accepted_ext PDU. The negotiation shall use the procedures defined in 9.3.6.2.5. See Sequence 67.



Sequence 67: Master rejects slave's request for an eSCO link.

9.3.6.2.3 Master or slave requests change of eSCO parameters

The master or slave may request a renegotiation of the eSCO parameters. The master or slave shall send an LMP_eSCO_link_req PDU with the eSCO handle of the eSCO link the device wishes to renegotiate. The remote device may accept the changed parameters immediately with LMP_accepted_ext PDU, or the negotiation may be continued with further LMP_eSCO_link_req PDUs until the master or slave accepts the latest parameters with an LMP_accepted_ext PDU or terminates the negotation with an LMP_not_accepted_ext PDU. In the case of termination with an LMP_not_accepted_ext PDU, the eSCO link continues on the previously negotiated parameters.

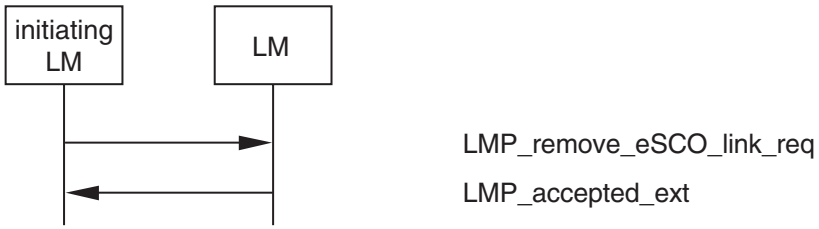
The sequence is the same as in 9.3.6.2.2.

During renegotiation, the eSCO LT_ADDR and eSCO handle shall not be renegotiated and shall be set to the originally negotiated values. The negotiation shall use the procedures defined in 9.3.6.2.5.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.6.2.4 Remove an eSCO link

Either the master or slave may remove the eSCO link by sending a request including the eSCO handle of the eSCO link to be removed and a error code indicating why the eSCO link is removed. The receiving side shall respond with an LMP_accepted_ext PDU. See Sequence 68.



Sequence 68: eSCO link removed.

9.3.6.2.5 Rules for the LMP negotiation and renegotiation

Rule 1: The negotiation state shall be set to 0 by the initiating LM. After the initial LMP_eSCO_link_req PDU is sent, the negotiation state shall not be set to 0.

Rule 2: If the bandwidth (defined as 1600 times the packet length in bytes divided by T_{eSCO} in slots) for either RX or TX or the air mode cannot be accepted, the device shall send an LMP_not_accepted_ext PDU with the appropriate error code.

Rule 3: Bandwidth and air mode are not negotiable and shall not be changed for the duration of the negotiation. Once one side has rejected the negotiation (with an LMP_not_accepted_ext PDU), a new negotiation may be started with different bandwidth and air mode parameters.

Rule 4: If the parameters would cause a latency violation (T_{eSCO} + W_{eSCO} + reserved synchronous slots > allowed local latency), the device should propose new parameters that shall not cause a reserved slot violation or latency violation for the device that is sending the parameters. In this case, the negotiation state shall be set to 3. Otherwise, the device shall send an LMP_not_accepted_ext PDU.

Rule 5: Once a device has received an LMP_eSCO_link_req PDU with the negotiation state set to 3 (latency violation), the device shall not propose any combination of packet type, T_{eSCO}, and W_{eSCO} that will give an equal or larger latency than the combination that caused the latency violation for the other device.

Rule 6: If the parameters would cause both a reserved slot violation and a latency violation, the device shall set the negotiation state to 3 (latency violation).

Rule 7: If the parameters would cause a reserved slot violation, the device should propose new parameters that shall not cause a reserved slot violation. In this case, the negotiation state shall be set to 2. Otherwise, the device shall send an LMP_not_accepted_ext PDU.

Rule 8: If the requested parameters are not supported, the device should propose a setting that is supported and set the negotiation state to 4. If it is not possible to find such a parameter set, the device shall send an LMP_not_accepted_ext PDU.

Rule 9: When proposing new parameters for reasons other than a latency violation, reserved slot violation, or configuration not supported, the negotiation state shall be set to 1.

9.3.6.2.6 Negotiation state definitions

The following terms are defined with regard to the negotiation state:

reserved slot violation: The receiving LM cannot set up the requested eSCO logical transport because the eSCO reserved slots would overlap with other regularly scheduled slots (e.g., other synchronous reserved slots, sniff instants, or park beacons).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

latency violation: The receiving LM cannot set up the requested eSCO logical transport because the latency (W_{eSCO} + T_{eSCO} + reserved synchronous slots) is greater than the maximum allowed latency.

configuration not supported: The combination of parameters requested is not inside the supported range for the device.

9.3.7 Test mode

This subclause describes the LMP procedures used to activate control and exit test mode. Throughout this subclause, the device that is placed in test mode is known as the *device under test (DUT)*.

9.3.7.1 Activation and deactivation of test mode

The activation may be carried out locally (via a hardware or software interface) or using the air interface.

— For activation over the air interface, entering the test mode shall be locally enabled for security and type approval reasons. The implementation of this local enabling is not subject to standardization.

  The tester sends an LMP command that shall force the DUT to enter test mode. The DUT shall terminate all normal operation before entering the test mode.

  If test mode is locally enabled, the DUT shall return an LMP_accepted PDU on reception of an activation command. An LMP_not_accepted PDU with the error code *PDU not allowed* shall be returned if the DUT is not locally enabled.

— If the activation is performed locally using a hardware or software interface, the DUT shall terminate all normal operation before entering the test mode.

  Until a connection to the tester exists, the device shall perform page scan and inquiry scan. Extended scan activity is recommended.

The DUT is always the slave.

See Sequence 69 and Sequence 70.



Sequence 69: Activation of test mode successful.



Sequence 70: Activation of test mode fails. Slave is not allowed to enter test mode.

The test mode can be deactivated in two ways. Sending an LMP_test_control PDU with the test scenario set to “exit test mode” exits the test mode, and the slave returns to normal operation still connected to the master. Sending an LMP_detach PDU to the DUT ends the test mode and the connection.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.3.7.2 Control of test mode

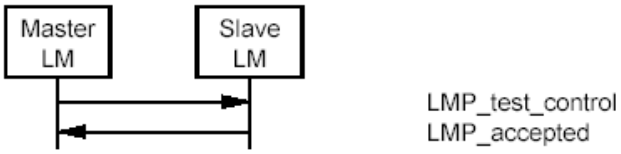
Control and configuration are performed using special LMP commands (see 9.3.7.3). These commands shall be rejected if the device is not in test mode. In this case, an LMP_not_accepted PDU shall be returned. The DUT shall return an LMP_accepted PDU on reception of a control command when in test mode.

An IEEE 802.15.1-2005 device in test mode shall ignore all LMP commands not related to control of the test mode. LMP commands dealing with power control and the request for LMP features (LMP_features_ req), and AFH (LMP_set_AFH, LMP_channel_classification_req, and LMP_channel_ classification) are allowed in test mode; the normal procedures are also used to test the adaptive power control.

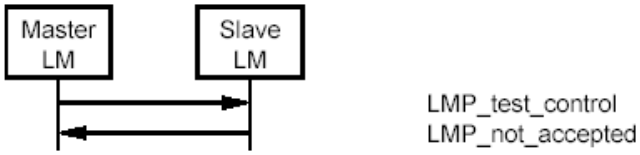
The DUT shall leave the test mode when an LMP_detach command is received or an LMP_test_control command is received with test scenario set to “exit test mode.”

When the DUT has entered test mode, the LMP_test_control PDU can be sent to the DUT to start a specific test. This PDU is acknowledged with an LMP_accepted PDU. If a device that is not in test mode receives an LMP_test_control PDU, it responds with an LMP_not_accepted PDU, where the error code shall be *PDU not allowed*.

See Sequence 71 and Sequence 72.



Sequence 71: Control of test mode successful.



Sequence 72: Control of test mode rejected since slave is not in test mode.

9.3.7.3 Summary of test mode PDUs

Table 64 lists all LMP messages used for test mode. To ensure that the contents of an LMP_test_control PDU are suitably whitened (important when sent in transmitter mode), each byte of all parameters listed in Table 65 are XORed with 0x55 before being sent.

The control PDU is used for both transmitter and loop back tests. The restrictions in Table 66 apply for the parameter settings.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 64—LMP messages used for test mode

LMP PDU	PDU number	Possible direction	Contents	Position in payload
LMP_test_activate	56	M → S		
LMP_test_control	57	M → S	Test scenario Hopping mode TX frequency RX frequency Power control mode Poll period Packet type Length of test data	2 3 4 5 6 7 8 9–10
LMP_detach	7	M → S		
LMP_accepted	3	M ← S		
LMP_not_accepted	4	M ← S		

Table 65—Parameters used in LMP_test_control PDU

Name	Length (bytes)	Type	Unit	Detailed
Test scenario	1	u_int8		0: Pause test mode 1: Transmitter test – 0 pattern 2: Transmitter test – 1 pattern 3: Transmitter test – 1010 pattern 4: Pseudorandom bit sequence 5: Closed loop back – ACL packets 6: Closed loop back – Synchronous packets 7: ACL packets without whitening 8: Synchronous packets without whitening 9: Transmitter test – 1111 0000 pattern 10–254: Reserved 255: Exit test mode The value is XORed with 0x55.
Hopping mode	1	u_int8		0: RX/TX on single frequency 1: Normal hopping 2: Reserved 3: Reserved 4: Reserved 5–255: Reserved The value is XORed with 0x55.
TX frequency (for DUT)	1	u_int8		f = [2402 + k] MHz The value is XORed with 0x55.
RX frequency (for DUT)	1	u_int8		f = [2402 + k] MHz The value is XORed with 0x55.
Power control mode	1	u_int8		0: Fixed TX output power 1: Adaptive power control The value is XORed with 0x55.
Poll period	1	u_int8	1.25 ms	The value is XORed with 0x55.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 65—Parameters used in LMP_test_control PDU *(continued)*

Name	Length (bytes)	Type	Unit	Detailed
Packet type	1	u_int8		Bits 3:0 Numbering as in packet header (see Clause 8) Bits 7:4 0: ACL/SCO 1: eSCO 2–15: Reserved The value is XORed with 0x55.
Length of test sequence (= length of user data in Clause 8)	2	u_int16	1 byte	Unsigned binary number The value is XORed with 0x55.

Table 66—Restrictions for parameters used in LMP_test_control PDU

Parameter	Restrictions transmitter test	Restrictions loopback test
TX frequency	$0 \le k \le 93$	$0 \le k \le 78$
RX frequency	Same as TX frequency	$0 \le k \le 78$
Poll period		Not applicable (set to 0)
Length of test sequence	Depends on packet type: **DH1**: ≤ 27 bytes **DH3**: ≤ 183 bytes **DH5**: ≤ 339 bytes **AUX1**: ≤ 29 bytes **HV3**: = 30 bytes **EV3**: ≤ 30 bytes **EV5**: ≤ 180 bytes	For ACL and SCO packets: not applicable (set to 0) For eSCO packets: **EV3**: ≤ 1–30 bytes **EV4**: ≤ 1–120 bytes **EV5**: ≤ 1–180 bytes

9.4 Summary

9.4.1 PDU summary

Table 67 shows the coding of the different LM PDUs.

Table 67—Coding of the different LM PDUs

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
Escape 1	Variable	124	**DM1**	M ↔ S	Extended opcode	2
					Variable	3–?
Escape 2	Variable	125	**DM1**	M ↔ S	Extended opcode	2
					Variable	3–?

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
Escape 3	Variable	126	**DM1**	M ↔ S	Extended opcode	2
					Variable	3–?
Escape 4	Variable	127	**DM1**	M ↔ S	Extended opcode	2
					Variable	3–?
LMP_accepted	2	3	**DM1**/ **DV**	M ↔ S	Opcode	2
LMP_accepted_ext	4	127/ 01	**DM1**	M ↔ S	Escape opcode	3
					Extended opcode	4
LMP_au_rand	17	11	**DM1**	M ↔ S	Random number	2–17
LMP_auto_rate	1	35	**DM1**/ **DV**	M ↔ S	—	
LMP_channel_ classification_req	7	127/ 16	**DM1**	M → S	AFH_reporting_mode	3
					AFH_min_interval	4–5
					AFH_max_interval	6–7
LMP_channel_ classification	12	127/ 17	**DM1**	M ← S	AFH_channel_ classification	3–12
LMP_clkoffset_req	1	5	**DM1**/ **DV**	M → S	—	
LMP_clkoffset_res	3	6	**DM1**/ **DV**	M ← S	Clock offset	2–3
LMP_comb_key	17	9	**DM1**	M ↔ S	Random number	2–17
LMP_decr_power_req	2	32	**DM1**/ **DV**	M ↔ S	For future use	2
LMP_detach	2	7	**DM1**/ **DV**	M ↔ S	Error code	2
LMP_encryption_key_ size_mask_req	1	58	**DM1**	M → S		
LMP_encryption_key_ size_mask_res	3	59	**DM1**	M ← S	Key size mask	2–3
LMP_encryption_key_ size_req	2	16	**DM1**/ **DV**	M ↔ S	Key size	2
LMP_encryption_mode_ req	2	15	**DM1**/ **DV**	M ↔ S	Encryption mode	2

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
LMP_eSCO_link_req	16	127/12	**DM1**	M ↔ S	eSCO handle	3
					eSCO LT_ADDR	4
					Timing control flags	5
					D_{eSCO}	6
					T_{eSCO}	7
					W_{eSCO}	8
					SCO packet type M→S	9
					SCO packet type S→M	10
					Packet length M→S	11–12
					Packet length S→M	13–14
					Air mode	15
					Negotiation state	16
LMP_features_req	9	39	**DM1**/**DV**	M ↔ S	Features	2–9
LMP_features_req_ext	12	127/03	**DM1**	M ↔ S	Features page	3
					Maximum supported page	4
					Extended features	5–12
LMP_features_res	9	40	**DM1**/**DV**	M ↔ S	Features	2–9
LMP_features_res_ext	12	127/04	**DM1**	M ↔ S	Features page	3
					Maximum supported page	4
					Extended features	5–12
LMP_host_connection_req	1	51	**DM1**/**DV**	M ↔ S	—	
LMP_hold	7	20	**DM1**/**DV**	M ↔ S	Hold time	2–3
					Hold instant	4–7
LMP_hold_req	7	21	**DM1**/**DV**	M ↔ S	Hold time	2–3
					Hold instant	4–7
LMP_incr_power_req	2	31	**DM1**/**DV**	M ↔ S	For future use	2
LMP_in_rand	17	8	**DM1**	M ↔ S	Random number	2–17
LMP_max_power	1	33	**DM1**/**DV**	M ↔ S	—	

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
LMP_max_slot	2	45	**DM1**/ **DV**	M ↔ S	Maximum slots	2
LMP_max_slot_req	2	46	**DM1**/ **DV**	M ↔ S	Maximum slots	2
LMP_min_power	1	34	**DM1**/ **DV**	M ↔ S	—	
LMP_modify_beacon	11 or 13	28	**DM1**	M → S	Timing control flags	2
					D_B	3–4
					T_B	5–6
					N_B	7
					Δ_B	8
					D_{access}	9
					T_{access}	10
					$N_{acc\text{-}slots}$	11
					N_{poll}	12
					M_{access}	13:0–3
					Access scheme	13:4–7
LMP_name_req	2	1	**DM1**/ **DV**	M ↔ S	Name offset	2
LMP_name_res	17	2	**DM1**	M ↔ S	Name offset	2
					Name length	3
					Name fragment	4–17
LMP_not_accepted	3	4	**DM1**/ **DV**	M ↔ S	Opcode	2
					Error code	3
LMP_not_accepted_ext	5	127/ 02	**DM1**	M ↔ S	Escape opcode	3
					Extended opcode	4
					Error code	5
LMP_page_mode_req	3	53	**DM1**/ **DV**	M ↔ S	Paging scheme	2
					Paging scheme settings	3
LMP_page_scan_mode_ req	3	54	**DM1**/ **DV**	M ↔ S	Paging scheme	2
					Paging scheme settings	3

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
LMP_park_req	17	25	**DM1**	M ↔ S	Timing control flags	2
					D_B	3–4
					T_B	5–6
					N_B	7
					Δ_B	8
					PM_ADDR	9
					AR_ADDR	10
					N_{Bsleep}	11
					D_{Bsleep}	12
					D_{access}	13
					T_{access}	14
					$N_{acc\text{-}slots}$	15
					N_{poll}	16
					M_{access}	17:0–3
					Access scheme	17:4–7
LMP_preferred_rate	2	36	**DM1**/ **DV**	M ↔ S	Data rate	2
LMP_quality_of_service	4	41	**DM1**/ **DV**	M → S	Poll interval	2–3
					N_{BC}	4
LMP_quality_of_service_req	4	42	**DM1**/ **DV**	M ↔ S	Poll interval	2–3
					N_{BC}	4
LMP_remove_eSCO_link_ req (see Note 4)	4	127/ 13	**DM1**	M ↔ S	eSCO handle	3
					Error code	4
LMP_remove_SCO_link_ req	3	44	**DM1**/ **DV**	M ↔ S	SCO handle	2
					Error code	3
LMP_SCO_link_req	7	43	**DM1**/ **DV**	M ↔ S	SCO handle	2
					Timing control flags	3
					D_{sco}	4
					T_{sco}	5
					SCO packet	6
					Air mode	7

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
LMP_set_AFH	16	60	**DM1**	M → S	AFH instant	2–5
					AFH mode	6
					AFH channel map	7–16
LMP_set_broadcast_ scan_window	4 or 6	27	**DM1**	M → S	Timing control flags	2
					D_B	3–4
					Broadcast scan window	5–6
LMP_setup_complete	1	49	**DM1**	M ↔ S	—	
LMP_slot_offset	9	52	**DM1**/ **DV**	M ↔ S	Slot offset	2–3
					BD_ADDR	4–9
LMP_sniff_req	10	23	**DM1**	M ↔ S	Timing control flags	2
					D_{sniff}	3–4
					T_{sniff}	5–6
					Sniff attempt	7–8
					Sniff timeout	9–10
LMP_sres	5	12	**DM1**/ **DV**	M ↔ S	Authentication response	2–5
LMP_start_encryption_req	17	17	**DM1**	M → S	Random number	2–17
LMP_stop_encryption_req	1	18	**DM1**/ **DV**	M → S	—	
LMP_supervision_timeout	3	55	**DM1**/ **DV**	M →s	Supervision timeout	2–3
LMP_switch_req	5	19	**DM1**/ **DV**	M ↔ S	Switch instant	2–5
LMP_temp_rand	17	13	**DM1**	M → S	Random number	2–17
LMP_temp_key	17	14	**DM1**	M → S	Key	2–17
LMP_test_activate	1	56	**DM1**/ **DV**	M → S	—	

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
LMP_test_control	10	57	**DM1**	M → S	Test scenario	2
					Hopping mode	3
					TX frequency	4
					RX frequency	5
					Power control mode	6
					Poll period	7
					Packet type	8
					Length of test data	9–10
LMP_timing_accuracy_req	1	47	**DM1**/ **DV**	M ↔ S	—	
LMP_timing_accuracy_res	3	48	**DM1**/ **DV**	M ↔ S	Drift	2
					Jitter	3
LMP_unit_key	17	10	**DM1**	M ↔ S	Key	2–17
LMP_unpark_BD_ADDR_ req	variable	29	**DM1**	M → S	Timing control flags	2
					D_B	3–4
					LT_ADDR 1st unpark	5:0–2
					LT_ADDR 2nd unpark	5:4–6
					BD_ADDR 1st unpark	6–11
					BD_ADDR 2nd unpark	12–17
LMP_unpark_PM_ADDR_ req	variable	30	**DM1**	M → S	Timing control flags	2
					D_B	3–4
					LT_ADDR 1st unpark	5:0–3
					LT_ADDR 2nd unpark	5:4–7
					PM_ADDR 1st unpark	6
					PM_ADDR 2nd unpark	7
					LT_ADDR 3rd unpark	8:0–3
					LT_ADDR 4th unpark	8:4–7
					PM_ADDR 3rd unpark	9
					PM_ADDR 4th unpark	10
					LT_ADDR 5th unpark	11:0–3
					LT_ADDR 6th unpark	11:4–7
					PM_ADDR 5th unpark	12

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	Op-code	Packet type	Possible direction	Contents	Position in payload
LMP_unpark_PM_ADDR_ req *(continued)*					PM_ADDR 6th unpark	13
					LT_ADDR 7th unpark	14:0–3
					PM_ADDR 7th unpark	15
LMP_unsniff_req	1	24	**DM1**/ **DV**	M ↔ S	—	
LMP_use_semi_ permanent_key	1	50	**DM1**/ **DV**	M → S	—	
LMP_version_req	6	37	**DM1**/ **DV**	M ↔ S	VersNr	2
					CompId	3–4
					SubVersNr	5–6
LMP_version_res	6	38	**DM1**/ **DV**	M ↔ S	VersNr	2
					CompId	3–4
					SubVersNr	5–6

NOTES

1—For LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_ADDR_req, and LMP_ unpark_PM_ADDR_req PDUs, the parameter D_B is optional. This parameter is present only if bit 0 of timing control flags is 1. If the parameter is not included, the position in payload for all parameters following D_B are decreased by 2.

2—For the LMP_unpark_BD_ADDR PDU, the LT_ADDR and the BD_ADDR of the 2nd unparked slave are optional. If only one slave is unparked, LT_ADDR 2nd unpark shall be zero, and BD_ADDR 2nd unpark is left out.

3—For the LMP_unpark_PM_ADDR PDU, the LT_ADDR and the PM_ADDR of the 2nd – 7th unparked slaves are optional. If N slaves are unparked, the fields up to and including the Nth unparked slave are present. If N is odd, the LT_ADDR $(N + 1)^{th}$ unpark shall be zero. The length of the message is $x + 3N/2$ if N is even and $x + 3(N + 1)/2 – 1$ if N is odd, where $x = 2$ or 4 depending on if the D_B is included or not (see Note 1).

4—Parameters coincide with their namesakes in LMP_<remove>_SCO_link_req PDUs apart from the following:

a) eSCO_LT_ADDR: The eSCO connection will be active on an additional LT_ADDR that needs to be defined. The master is allowed to reassign an active eSCO link to a different LT_ADDR.
b) D_{eSCO}, T_{eSCO}: As per LMP_SCO_link_req, but with a greater flexibility in values (e.g., no longer fixed with respect to **HV1**, **HV2**, and **HV3** packet choice).
c) W_{eSCO}: The eSCO retransmission window size (in slots).
d) Packet type and packet length may be set differently in master-to-slave or slave-to-master directions for asynchronous eSCO links.
e) Packet length (in bytes): eSCO packet types no longer have fixed length
f) Negotiation state: This is used to better enable the negotiation of the negotiable parameters: D_{eSCO}, T_{eSCO}, W_{eSCO}, eSCO packet type M→S, eSCO packet type S→M, packet length M→S, packet length S→M. When responding to an eSCO link request with a new suggestion for these parameters, this flag may be set to 1 to indicate that the last received negotiable parameters are possible, but the new parameters specified in the response eSCO link request would be preferable, to 2 to indicate that the last received negotiable parameters are not possible as they cause a reserved slot violation, or to 3 to indicate that the last received negotiable parameters would cause a latency violation. The flag shall be set to zero in the initiating LMP_eSCO_ link_req PDU.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.4.2 Parameter definitions

The parameter definitions are listed in Table 68.

Table 68—Parameters in LM PDUs

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
Access scheme	1	u_int4		0: Polling technique 1–15: Reserved	
AFH_channel_ classification	10	multiple bytes	—	This parameter contains 40 two-bit fields. The nth (numbering from 0) such field defines the classification of channels $2n$ and $2n + 1$, other than the 39th field, which just contains the classification of channel 78. Each field interpreted as an integer whose values indicate: 0 = unknown 1 = good 2 = reserved 3 = bad	
AFH_channel_map	10	multiple bytes	—	If AFH_mode is AFH_enabled, this parameter contains 79 one-bit fields; otherwise, the contents are reserved. The nth (numbering from 0) such field (in the range 0 to 78) contains the value for channel n. Bit 79 is reserved (set to 0 when transmitted and ignored when received). The 1-bit field is interpreted as follows: 0: channel n is unused 1: channel n is used	
AFH_instant	4	u_int32	slots	Bits 27:1 of the CLK value at the time of switching hop sequences. Must be even.	
AFH_max_interval	2	u_int16	slots	Range is 0x0640 to 0xBB80 slots (1 to 30 s)	
AFH_min_interval	2	u_int16	slots	Range is 0x0640 to 0xBB80 slots (1 to 30 s)	
AFH_mode	1	u_int8	—	0: AFH disabled 1: AFH enabled 2–255: Reserved	
AFH_reporting_ mode	1	u_int8	—	0: AFH reporting disabled 1: AFH reporting enabled 2–255: Reserved	

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 68—Parameters in LM PDUs *(continued)*

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
Air mode	1	u_int8		0: μ-law log 1: A-law log 2: CVSD 3: Transparent data 4–255: Reserved	See Table 69
AR_ADDR	1	u_int8			
Authentication response	4	multiple bytes			
BD_ADDR	6	multiple bytes		BD_ADDR of the sending device	
Broadcast scan window	2	u_int16	slots		
Clock offset	2	u_int16	1.25 ms	($CLKN_{16-2}$ slave – $CLKN_{16-2}$ master) mod 2^{15} MSB of second byte not used.	
CompId	2	u_int16		See Bluetooth Assigned Numbers [B1]	
D_{access}	1	u_int8	slots		
Data rate	1	u_int8		Bit 0 = 0: Use FEC Bit 0 = 1: Do not use FEC Bit 1–2 = 0: No packet-size preference available Bit 1–2 = 1: Use 1-slot packets Bit 1–2 = 2: Use 3-slot packets Bit 1–2 = 3: Use 5-slot packets Bit 3–7: Reserved	
D_B	2	u_int16	slots		
Δ_B	1	u_int8	slots		
D_{Bsleep}	1	u_int8			
D_{eSCO}	1	u_int8	slots	Valid range is 0–254 slots	See Table 69.
Drift	1	u_int8	ppm		
D_{sco}	1	u_int8	slots	Only even values are valid[a]	0 to (T_{sco} – 2)
D_{sniff}	2	u_int16	slots	Only even values are valid[a]	0 to (T_{sniff} – 2)
Encryption mode	1	u_int8		0: No encryption 1: Encryption 2: Encryption 3–255: Reserved	
Error code	1	u_int8		See 10.3	
Escape opcode	1	u_int8		Identifies which escape opcode is being acknowledged: range 124–127	
eSCO handle	1	u_int8			

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 68—Parameters in LM PDUs *(continued)*

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
eSCO LT_ADDR	1	u_int8		LT_ADDR for the eSCO logical transport. The range is extended to 8 bits compared with the normal LT_ADDR field: range 0–7.	0–7
eSCO packet type	1	u_int8		0x00: NULL/POLL 0x07: **EV3** 0x0C: **EV4** 0x0D: **EV5** Other values are reserved	If the value is 0x00, the POLL packet shall be used by the master, and the NULL packet shall be used by the slave. See Table 69.
Extended features	8	multiple bytes		One page of extended features	
Extended opcode	1	u_int8		Which extended opcode is being acknowledged	
Features	8	multiple bytes		See Table 35.	
Features page	1	u_int8		Identifies which page of extended features is being requested. 0 means standard features 1–255 other feature pages	
Hold instant	4	u_int32	slots	Bits 27:1 of the CLK value	
Hold time	2	u_int16	slots	Only even values are valid[1]	0x0014–0x8000; shall not exceed (*supervisionTO* * 0.999)
Jitter	1	u_int8	μs		
Key	16	multiple bytes			
Key size	1	u_int8	byte		
Key size mask	2	u_int16		Bit mask of supported broadcast encryption key sizes: LSB is support for length 1, and so on. The bit shall be one if the key size is supported.	
LT_ADDR	1	u_int4			
M_{access}	1	u_int4		Number of access windows	
Max slots	1	u_int8	slots		
Max supported page	1	u_int8		Highest page of extended features that contains a nonzero bit for the originating device. Range 0–255	

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 68—Parameters in LM PDUs *(continued)*

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
$N_{acc\text{-}slots}$	1	u_int8	slots		
Name fragment	14	multiple bytes		UTF-8 characters.	
Name length	1	u_int8	bytes		
Name offset	1	u_int8	bytes		
N_B	1	u_int8			
N_{BC}	1	u_int8			
N_{Bsleep}	1	u_int8			
Negotiation state	1	u_int8		0: Initiate negotiation 1: The latest received set of negotiable parameters were possible, but these parameters are preferred. 2: The latest received set of negotiable parameters would cause a reserved slot violation. 3: The latest received set of negotiable parameters would cause a latency violation. 4: The latest received set of egotiable parameters are not supported. Other values are reserved.	
N_{poll}	1	u_int8			
Opcode	1	u_int8			
Packet length	2	u_int16	bytes	Length of the eSCO payload 0 for POLL/NULL 1–30 for **EV3** 1–120 for **EV4** 1–180 for **EV5** Other values are invalid	See Table 69.
Paging scheme	1	u_int8		0: Mandatory scheme 1-255: Reserved	
Paging scheme settings	1	u_int8		For mandatory scheme: 0: R0 1: R1 2: R2 3–255: Reserved	
PM_ADDR	1	u_int8			
Poll interval	2	u_int16	slots	Only even values are valid[a]	0x0006–0x1000
Random number	16	multiple bytes			

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 68—Parameters in LM PDUs *(continued)*

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
Reserved(*n*)	*n*	u_int8		Reserved for future use – must be 0 when transmitted, ignore value when received	
SCO handle	1	u_int8			
SCO packet	1	u_int8		0: **HV1** 1: **HV2** 2: **HV3** 3–255: Reserved	
Slot offset	2	u_int16	μs	0 ≤ slot offset < 1250	
Sniff attempt	2	u_int16	received slots	Number of receive slots	1 to $T_{sniff}/2$
Sniff timeout	2	u_int16	received slots	Number of receive slots	0–0x0028
SubVersNr	2	u_int16		Defined by each company	
Supervision timeout	2	u_int16	slots	0 means an infinite timeout	0 and 0x0190–0xFFFF
Switch instant	4	u_int32	slots	Bits 27:1 of the CLK value	
T_{access}	1	u_int8	slots		
T_B	2	u_int16	slots		
T_{eSCO}	1	u_int8	slots	Valid range is 4–254 slots	See Table 69
Timing control flags	1	u_int8		Bit 0 = 0: No timing change Bit 0 = 1: Timing change Bit 1 = 0: Use initialization 1 Bit 1 = 1: Use initialization 2 Bit 2 = 0: Access window Bit 2 = 1: No access window Bit 3–7: Reserved	
T_{sco}	1	u_int8	slots	Only even values are valid[a]	2–6
T_{sniff}	2	u_int16	slots	Only even values are valid[a]	0x0006–0x0540; shall not exceed (*supervisionTO* * 0.999)
VersNr	1	u_int8		See Bluetooth Assigned Numbers [B1]	
W_{eSCO}	1	u_int8	slots	Number of slots in the retransmission window. Valid range is 0–254 slots.	See Table 69.

[a]If a device receives an LMP PDU with an odd value in this parameter field, the PDU should be rejected with an error code of *invalid LMP parameters*.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

9.4.3 Default values

Devices shall use the values in Table 69 before anything else has been negotiated.

Table 69—Mandatory parameter ranges for eSCO packet types

Parameter	EV3	EV4	EV5
D_{eSCO}	0–4 (even)	0–14 (even)	0–14 (even)
T_{eSCO}	6	16 (even)	16 (even)
W_{eSCO}	0–4 (even)	0–6 (even)	0–6 (even)
eSCO packet type M→S	**EV3**	**EV3**, **EV4**	**EV3**, **EV5**
eSCO packet type S→M	**EV3**	**EV3**, **EV4**	**EV3**, **EV5**
Packet length M→S	30	1–120	1–180
Packet length S→M	30	1–120	1–180
Air mode	At least one of A-law, μ-law, CVSD, transparent	Transparent	Transparent

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10. Error codes

This clause lists the various possible error codes. When a command fails or an LMP message needs to indicate a failure, error codes are used to indicate the reason for the error. Error codes have a size of one octet.

The purpose of this clause is to give descriptions of how the error codes should be used. It is beyond the scope of this standard to give detailed descriptions of all situations where error codes can be used, especially as this may be implementation dependent.

10.1 HCI command errors

If an HCI command that should generate an HCI_Command_Complete event generates an error, then this error shall be reported in the HCI_Command_Complete event.

If an HCI command that sent an HCI_Command_Status with the error code *success* to the host before processing finds an error during execution, then the error may be reported in the normal completion command for the original command or in an HCI_Command_Status event.

Some HCI commands may generate errors that need to be reported to be host, but there is insufficient information to determine how the command would normally be processed. In this case, two events can be used to indicate this to the host, the HCI_Command_Complete event and HCI_Command_Status event. Which of the two events is used is implementation dependent.

10.2 List of error codes

The error code of 0x00 means *success*. The possible range of failure error codes is 0x01–0xFF. Subclause 10.3 provides an error code usage description for each failure error code.

Table 70—List of possible error codes

Error code	Name
0x00	Success
0x01	Unknown HCI command
0x02	Unknown connection identifier
0x03	Hardware failure
0x04	Page timeout
0x05	Authentication failure
0x06	PIN missing
0x07	Memory capacity exceeded
0x08	Connection timeout
0x09	Connection limit exceeded
0x0A	Synchronous connection limit to a device exceeded
0x0B	ACL connection already exists
0x0C	Command disallowed

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 70—List of possible error codes *(continued)*

Error code	Name
0x0D	Connection rejected due to limited resources
0x0E	Connection rejected due to security reasons
0x0F	Connection rejected due to unacceptable BD_ADDR
0x10	Connection accept timeout exceeded
0x11	Unsupported feature or parameter value
0x12	Invalid HCI command parameters
0x13	Remote user terminated connection
0x14	Remote device terminated connection due to low resources
0x15	Remote device terminated connection due to power off
0x16	Connection terminated by local host
0x17	Repeated attempts
0x18	Pairing not allowed
0x19	Unknown LMP PDU
0x1A	Unsupported remote feature
0x1B	SCO offset rejected
0x1C	SCO interval rejected
0x1D	SCO air mode rejected
0x1E	Invalid LMP parameters
0x1F	Unspecified error
0x20	Unsupported LMP parameter value
0x21	Role change not allowed
0x22	LMP response timeout
0x23	LMP error transaction collision
0x24	LMP PDU not allowed
0x25	Encryption mode not acceptable
0x26	Link key cannot be changed
0x27	Requested QoS not supported
0x28	Instant passed
0x29	Pairing with unit key not supported
0x2A	Different transaction collision
0x2B	Reserved
0x2C	QoS unacceptable parameter
0x2D	QoS rejected

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 70—List of possible error codes *(continued)*

Error code	Name
0x2E	Channel classification not supported
0x2F	Insufficient security
0x30	Parameter out of mandatory range
0x31	Reserved
0x32	Role switch pending
0x33	Reserved
0x34	Reserved slot violation
0x35	Role switch failed

10.3 Error code descriptions

This subclause provides detailed descriptions of the error codes and examples of their usage.

10.3.1 Unknown HCI command (0x01)

The error code *unknown HCI command* indicates that the controller does not understand the HCI command packet opcode that the host sent. The opcode given might not correspond to any of the opcodes specified in this standard or any vendor-specific opcodes, or the command may not have been implemented.

10.3.2 Unknown connection identifier (0x02)

The error code *unknown connection identifier* indicates that a command was sent from the host that should identify a connection, but that connection does not exist.

10.3.3 Hardware failure (0x03)

The error code *hardware failure* indicates to the host that something in the controller has failed in a manner that cannot be described with any other error code. The meaning implied with this error code is implementation dependent.

10.3.4 Page timeout (0x04)

The error code *page timeout* indicates that a page timed out because of the page timeout configuration parameter. This error code may occur only with the HCI_Remote_Name_Request and HCI_Create_Connection commands.

10.3.5 Authentication failure (0x05)

The error code *authentication failure* indicates that pairing or authentication failed due to incorrect results in the pairing or authentication procedure. This could be due to an incorrect PIN or link key.

10.3.6 PIN missing (0x06)

The error code *PIN missing* is used when pairing failed because of a missing PIN.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10.3.7 Memory capacity exceeded (0x07)

The error code *memory capacity exceeded* indicates to the host that the controller has run out of memory to store new parameters.

10.3.8 Connection timeout (0x08)

The error code *connection timeout* indicates that the link supervision timeout has expired for a given connection.

10.3.9 Connection limit exceeded (0x09)

The error code *connection limit exceeded* indicates that an attempt to create another connection failed because the controller is already at its limit of the number of connections it can support. The number of connections a device can support is implementation dependent.

10.3.10 Synchronous connection limit to a device exceeded (0x0A)

The error code *synchronous connection limit to a device exceeded* indicates that the controller has reached the limit to the number of synchronous connections that can be achieved to a device. The number of synchronous connections a device can support is implementation dependent.

10.3.11 ACL connection already exists (0x0B)

The error code *ACL connection already exists* indicates that an attempt to create a new ACL connection to a device when there is already a connection to this device.

10.3.12 Command disallowed (0x0C)

The error code *command disallowed* indicates that the command requested cannot be executed because the controller is in a state where it cannot process this command at this time. This error shall not be used for command opcodes where the error code *unknown HCI command* is valid.

10.3.13 Connection rejected due to limited resources (0x0D)

The error code *connection rejected due to limited resources* indicates that an incoming connection was rejected due to limited resources.

10.3.14 Connection rejected due to security reasons (0x0E)

The error code *connection rejected due to security reasons* indicates that a connection was rejected due to security requirements not being fulfilled, e.g., authentication or pairing.

10.3.15 Connection rejected due to unacceptable BD_ADDR (0x0F)

The error code *connection rejected due to unacceptable BD_ADDR* indicates that a connection was rejected because this device does not accept the BD_ADDR. This may be because the device will accept connections only from specific BD_ADDRs.

10.3.16 Connection accept timeout exceeded (0x10)

The error code *connection accept timeout exceeded* indicates that the connection accept timeout has been exceeded for this connection attempt.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10.3.17 Unsupported feature or parameter value (0x11)

The error code *unsupported feature or parameter value* indicates that a feature or parameter value in an LMP message or HCI command is not supported.

10.3.18 Invalid HCI command parameters (0x12)

The error code *invalid HCI command parameters* indicates that at least one of the HCI command parameters is invalid. This shall be used when

— The parameter total length is invalid.
— A command parameter is an invalid type.
— A connection identifier does not match the corresponding event.
— A parameter value must be even.
— A parameter is outside of the specified range.
— Two or more parameter values have inconsistent values.

NOTE—An invalid type can be, for example, when an SCO connection handle is used where an ACL connection handle is required.

10.3.19 Remote user terminated connection (0x13)

The error code *Remote User Terminated Connection* indicates that the user on the remote device terminated the connection.

10.3.20 Remote device terminated connection due to low resources (0x14)

The error code *remote device terminated connection due to low resources* indicates that the remote device terminated the connection because of low resources.

10.3.21 Remote device terminated connection due to power off (0x15)

The error code *remote device terminated connection due to power off* indicates that the remote device terminated the connection because the device is about to power off.

10.3.22 Connection terminated by local host (0x16)

The error code *connection terminated by local host* indicates that the local device terminated the connection.

10.3.23 Repeated attempts (0x17)

The error code *repeated attempts* indicates that the controller is disallowing an authentication or pairing procedure because too little time has elapsed since the last authentication or pairing attempt failed.

10.3.24 Pairing not allowed (0x18)

The error code *pairing not allowed* indicates that the device does not allow pairing. For example, a device may allow pairing only during a certain time window after some user input allows pairing.

10.3.25 Unknown LMP PDU (0x19)

The error code *unknown LMP PDU* indicates that the controller has received an unknown LMP opcode.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10.3.26 Unsupported remote feature (0x1A)

The error code *unsupported remote feature* indicates that the remote device does not support the feature associated with the issued command or LMP PDU.

10.3.27 SCO offset rejected (0x1B)

The error code *SCO offset rejected* indicates that the offset requested in the LMP_SCO_link_req message has been rejected.

10.3.28 SCO interval rejected (0x1C)

The error code *SCO interval rejected* indicates that the interval requested in the LMP_SCO_link_req message has been rejected.

10.3.29 SCO air mode rejected (0x1D)

The error code *SCO air mode rejected* indicates that the air mode requested in the LMP_SCO_link_req message has been rejected.

10.3.30 Invalid LMP parameters (0x1E)

The error code *invalid LMP parameters* indicates that some LMP message parameters were invalid. This shall be used when

— The PDU length is invalid.
— A parameter value must be even.
— A parameter is outside of the specified range.
— Two or more parameters have inconsistent values.

10.3.31 Unspecified error (0x1F)

The error code unspecified error indicates that no other error code specified is appropriate to use.

10.3.32 Unsupported LMP parameter value (0x20)

The error code *unsupported LMP parameter value* indicates that an LMP message contains at least one parameter value that is not supported by the controller at this time. This is normally used after a long negotiation procedure, e.g., during an LMP_hold_req, LMP_sniff_req, and LMP_encryption_key_size_req message exchanges.

10.3.33 Role change not allowed (0x21)

The error code *role change not allowed* indicates that a controller will not allow a role change at this time.

10.3.34 LMP response timeout (0x22)

The error code *LMP response timeout* indicates that an LMP transaction failed to respond within the LMP response timeout.

10.3.35 LMP error transaction collision (0x23)

The error code *LMP error transaction collision* indicates that an LMP transaction has collided with the same transaction that is already in progress.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10.3.36 LMP PDU not allowed (0x24)

The error code *LMP PDU not allowed* indicates that a controller sent an LMP message with an opcode that was not allowed.

10.3.37 Encryption mode not acceptable (0x25)

The error code *encryption mode not acceptable* indicates that the requested encryption mode is not acceptable at this time.

10.3.38 Link key cannot be changed (0x26)

The error code *link key cannot be changed* indicates that a link key cannot be changed because a fixed unit key is being used.

10.3.39 Requested QoS not supported (0x27)

The error code *requested QoS not supported* indicates that the requested QoS is not supported.

10.3.40 Instant passed (0x28)

The error code *instant passed* indicates that an LMP PDU that includes an instant cannot be performed because the instant when this would have occurred has passed.

10.3.41 Pairing with unit key not supported (0x29)

The error code *pairing with unit key not supported* indicates that it was not possible to pair as a unit key was requested and it is not supported.

10.3.42 Different transaction collision (0x2A)

The error code *different transaction collision* indicates that an LMP transaction was started that collides with an ongoing transaction.

10.3.43 QoS unacceptable parameter (0x2C)

The error code *QoS unacceptable parameter* indicates that the specified QoS parameters could not be accepted at this time, but other parameters may be acceptable.

10.3.44 QoS rejected (0x2D)

The error code *QoS rejected* indicates that the specified QoS parameters cannot be accepted and QoS negotiation should be terminated.

10.3.45 Channel classification not supported (0x2E)

The error code *channel classification not supported* indicates that the controller cannot perform channel classification because it is not supported.

10.3.46 Insufficient security (0x2F)

The error code *insufficient security* indicates that the HCI command or LMP message sent is possible only on an encrypted link.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

10.3.47 Parameter out of mandatory range (0x30)

The error code *parameter out of mandatory range* indicates that a parameter value requested is outside the mandatory range of parameters for the given HCI command or LMP message.

10.3.48 Role switch pending (0x32)

The error code *role switch pending* indicates that a role switch is pending. This can be used when an HCI command or LMP message cannot be accepted because of a pending role switch. This can also be used to notify a peer device about a pending role switch.

10.3.49 Reserved slot violation (0x34)

The error code *reserved slot violation* indicates that the current synchronous negotiation was terminated with the negotiation state set to reserved slot violation.

10.3.50 Role switch failed (0x35)

The error code *role switch failed* indicates that a role switch was attempted, but it failed and the original piconet structure is restored. The switch may have failed because the TDD switch or piconet switch failed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11. Host controller interface (HCI)

This clause describes the HCI, which provides a command interface to the BB controller and LM and provides access to configuration parameters. The HCI provides a uniform interface method of accessing the controller capabilities. Subclause 11.1 provides a brief overview of the lower layers of the software stack and of the hardware. Subclause 11.2 provides an overview of the host controller transport. Subclause 11.3 provides an overview of the commands and events. Subclause 11.4 describes the flow control used between the host and the controller. Subclause 11.5 describes the HCI data formats, and subclause 11.6 describes the HCI configuration parameters. Subclause 11.7 describes each of the HCI commands in details, identifies parameters for each of the commands, and lists events associated with each command. Subclause 11.8 gives the commands, events, and configuration parameters that are now deprecated.

11.1 Lower layers of the IEEE 802.15.1-2005 software stack

Figure 91 provides an overview of the lower software layers. The HCI firmware implements the HCI commands for the IEEE 802.15.1-2005 hardware by accessing BB commands, LM commands, hardware status registers, control registers, and event registers.

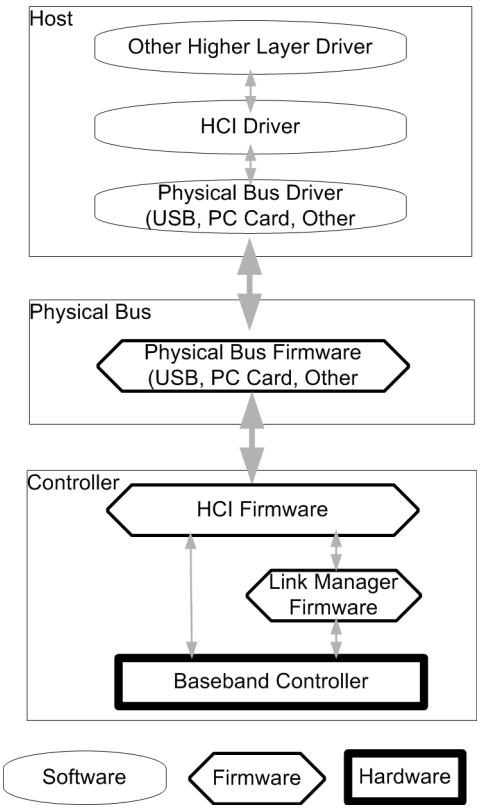


Figure 91—Overview of the lower software layers

Several layers may exist between the HCI driver on the host system and the HCI firmware in the IEEE 802.15.1-2005 hardware. These intermediate layers, the host controller transport layer, provide the ability to transfer data without knowledge of the data contents or format.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Figure 92 illustrates the path of a data transfer from one device to another. The HCI driver on the host exchanges data and commands with the HCI firmware on the hardware. The host controller transport layer (i.e., physical bus) driver provides both HCI layers with the ability to exchange information with each other.

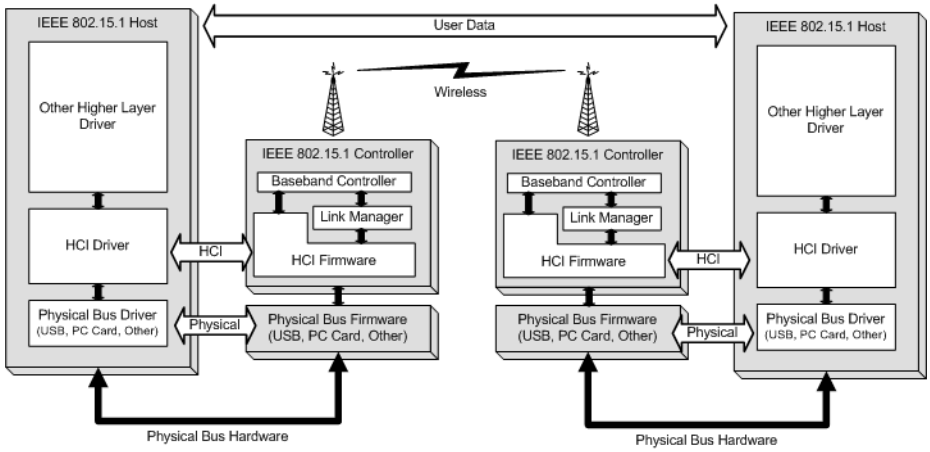


Figure 92—End-to-end overview of lower software layers to transfer data

The host will receive asynchronous notifications of HCI events independent of which host controller transport layer is used. HCI events are used for notifying the host when something occurs. When the host discovers that an event has occurred, it will then parse the received event packet to determine which event occurred.

11.2 Overview of host controller transport

The host driver stack has a transport layer between the host controller driver and the host.

The main goal of this transport layer is transparency. The host controller driver (which interfaces to the controller) should be independent of the underlying transport technology. Nor should the transport require any visibility into the data that the host controller driver passes to the controller. This allows the interface (HCI) or the controller to be upgraded without affecting the transport layer.

The specified host controller transport layers are described in a separate volume (see Bluetooth specification volume 4).

11.3 Overview of commands and events

The commands and events are sent between the host and the controller. These are grouped into logical groups. The version information in this subclause denotes the version number of the Bluetooth specification in which a command or event was first specified.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.3.1 Generic events

The generic events occur due to multiple commands or due to events that can occur at any time. See Table 71.

Table 71—Generic events

Name	Version	Summary description
Command Complete event	1.1	Used by the controller to pass the return status of a command and the other event parameters for each HCI command.
Command Status event	1.1	Indicates that the command described by the Command_Opcode parameter has been received and the controller is currently performing the task for this command.
Hardware Error event	1.1	Indicates some type of hardware failure for the controller.

11.3.2 Device setup

The device setup group of commands is used to place the controller into a known state. See Table 72.

Table 72—Device setup

Name	Version	Summary description
Reset command	1.1	Resets the controller, LM, and the PHY.

11.3.3 Controller flow control

The controller flow control group of commands and events is used to control data flow from the host to the controller. See Table 73.

Table 73—Controller flow control

Name	Version	Summary description
Read Buffer Size command	1.1	Returns the size of the HCI buffers. These buffers are used by the controller to buffer data that are to be transmitted.
Number Of Completed Packets event	1.1	Used by the controller to indicate to the host how many HCI data packets have been completed for each connection handle since the previous Number Of Completed Packets event was sent.

11.3.4 Controller information

The controller information group of commands allows the host to discover local information about the device. See Table 74.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 74—Controller information

Name	Version	Summary description
Read Local Version Information command	1.1	Reads the version information for the local device.
Read Local Supported Commands command	1.2	Requests a list of the supported HCI commands for the local device.
Read Local Supported Features command	1.1	Requests a list of the supported features for the local device.
Read Local Extended Features command	1.2	Requests a list of the supported extended features for the local device
Read BD_ADDR command	1.1	Reads the value for the BD_ADDR parameter.

11.3.5 Controller configuration

The controller configuration group of commands and events allows the global configuration parameters to be configured. Table 75 shows the commands and associated version numbers for the configuration process.

Table 75—Controller configuration

Name	Version	Summary description
Read Local Name command	1.1	Provides the ability to read the stored user-friendly name for the device.
Write Local Name command	1.1	Provides the ability to modify the user-friendly name for the device.
Read Class Of Device command	1.1	Reads the value for the Class_of_Device configuration parameter, which is used to indicate its capabilities to other devices.
Write Class Of Device command	1.1	Writes the value for the Class_of_Device configuration parameter, which is used to indicate its capabilities to other devices.
Read Number Of Supported IAC command	1.1	Reads the value for the number of IACs for which the local device can simultaneously listen during an inquiry scan.
Read Current IAC LAP command	1.1	Reads the LAP(s) used to create the IACs for which the local device is simultaneously scanning during inquiry scans.
Write Current IAC LAP command	1.1	Writes the LAP(s) used to create the IACs for which the local device is simultaneously scanning during inquiry scans.
Read Scan Enable command	1.1	Reads the value for the Scan_Enable configuration parameter, which controls whether the device will periodically scan for page attempts and/or inquiry requests from other devices.
Write Scan Enable command	1.1	Writes the value for the Scan_Enable configuration parameter, which controls whether the device will periodically scan for page attempts and/or inquiry requests from other devices.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.3.6 Device discovery

The device discovery group of commands and events allows a device to discover other devices in the surrounding area. Table 76 shows the commands and associated version numbers for device discovery.

Table 76—Device discovery

Name	Version	Summary description
Inquiry command	1.1	Causes the device to enter inquiry mode. Inquiry mode is used to discovery other nearby devices.
Inquiry Result event	1.1	Indicates that a device or multiple devices have responded so far during the current inquiry process.
Inquiry Result With RSSI event	1.2	Indicates that a device or multiple devices have responded so far during the current inquiry process.
Inquiry Cancel command	1.1	Causes the device to stop the current inquiry if the device is in inquiry mode.
Inquiry Complete event	1.1	Indicates that the inquiry is finished.
Periodic Inquiry Mode command	1.1	Configures the device to perform an automatic inquiry based on a specified period range.
Exit Periodic Inquiry Mode command	1.1	Ends the periodic inquiry mode when the local device is in periodic inquiry mode.
Read Inquiry Scan Activity command	1.1	Reads the value for Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameters. Inquiry scan interval defines the amount of time between consecutive inquiry scans. Inquiry scan window defines the amount of time for the duration of the inquiry scan.
Write Inquiry Scan Activity command	1.1	Writes the value for Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameters. Inquiry scan interval defines the amount of time between consecutive inquiry scans. Inquiry scan window defines the amount of time for the duration of the inquiry scan.
Read Inquiry Scan Type command	1.2	Reads the Inquiry_Scan_Type configuration parameter of the local device. The Inquiry_Scan_Type configuration parameter can set the inquiry scan to either normal or interlaced scan.
Write Inquiry Scan Type command	1.2	Writes the Inquiry_Scan_Type configuration parameter of the local device. The Inquiry_Scan_Type configuration parameter can set the inquiry scan to either normal or interlaced scan.
Read Inquiry Mode command	1.2	Reads the Inquiry_Mode configuration parameter of the local device.
Write Inquiry Mode command	1.2	Writes the Inquiry_Mode configuration parameter of the local device.

11.3.7 Connection setup

The connection setup group of commands and events is used to allow a device to make a connection to another device. See Table 77.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 77—Connection setup

Name	Version	Summary description
Create Connection command	1.1	Causes the LM to create an ACL connection to the device with the BD_ADDR specified by the command parameters.
Connection Request event	1.1	Indicates that a new incoming connection is trying to be established.
Accept Connection Request command	1.1	Accepts a new incoming connection request.
Reject Connection Request command	1.1	Rejects a new incoming connection request.
Create Connection Cancel command	1.2	Cancels an ongoing connection creation.
Connection Complete event	1.1	Indicates to both of the hosts forming the connection that a new connection has been established.
Disconnect command	1.1	Terminates an existing connection.
Disconnection Complete event	1.1	Occurs when a connection has been terminated.
Read Page Timeout command	1.1	Reads the value for the Page_Reply_Timeout configuration parameter, which determines the time the controller will wait for the remote device to respond to a connection request before the local device returns a connection failure.
Write Page Timeout command	1.1	Writes the value for the Page_Reply_Timeout configuration parameter, which allows the IEEE 802.15.1-2005 hardware to define the amount of time a connection request will wait for the remote device to respond before the local device returns a connection failure.
Read Page Scan Activity command	1.1	Reads the values for the Page_Scan_Interval and Page_Scan_ Window configuration parameters. Page scan interval defines the amount of time between consecutive page scans. Page scan window defines the duration of the page scan.
Write Page Scan Activity command	1.1	Writes the values for Page_Scan_Interval and Page_Scan_ Window configuration parameters. Page scan interval defines the amount of time between consecutive page scans. Page scan window defines the duration of the page scan.
Page Scan Repetition Mode Change event	1.1	Indicates that the connected remote device with the specified connection handle has successfully changed the page scan repetition mode.
Read Page Scan Type command	1.2	Reads the page scan type of the local device. The Page_Scan_ Type configuration parameter has two possible values: normal scan or interlaced scan.
Write Page Scan Type command	1.2	Writes the page scan type of the local device. The Page_Scan_ Type configuration parameter can set the page scan to either normal or interlaced scan.
Read Connection Accept Timeout command	1.1	Reads the value for the Connection_Accept_Timeout configuration parameter, which allows the hardware to automatically deny a connection request after a specified period has occurred and to refuse a new connection.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 77—Connection setup *(continued)*

Name	Version	Summary description
Write Connection Accept Timeout command	1.1	Writes the value for the Connection_Accept_Timeout configuration parameter, which allows the IEEE 802.15.1-2005 hardware to automatically deny a connection request after a specified period has occurred and to refuse a new connection.
Read Page Scan Period Mode command	1.1	Reads the mandatory page scan period mode of the local device.
Write Page Scan Period Mode command	1.1	Writes the mandatory page scan period mode of the local device.

11.3.8 Remote information

The remote information group of commands and events allows information about a remote devices configuration to be discovered. See Table 78.

Table 78—Remote information

Name	Version	Summary description
Remote Name Request command	1.1	Obtains the user-friendly name of another device.
Remote Name Request Cancel command	1.2	Cancel an ongoing remote name request.
Remote Name Request Complete event	1.1	Indicates a remote name request has been completed.
Read Remote Supported Features command	1.1	Requests a list of the supported features of a remote device.
Read Remote Supported Features Complete event	1.1	Indicates the completion of the process where the LM obtains the supported features of the remote device specified by the connection handle event parameter.
Read Remote Extended Features command	1.2	Requests a list of the supported extended features of a remote device
Read Remote Extended Features Complete event	1.2	Indicates the completion of the process where the LM obtains the supported extended features of the remote device specified by the connection handle event parameter.
Read Remote Version Information command	1.1	Reads the values for the version information for the remote device.
Read Remote Version Information Complete event	1.1	Indicates the completion of the process where the LM obtains the version information of the remote device specified by the connection handle event parameter.

11.3.9 Synchronous connections

The synchronous connections group of commands and events allows synchronous connections to be created. See Table 79.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 79—Synchronous connections

Name	Version	Summary description
Setup Synchronous Connection command	1.2	Adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the connection handle parameter specified.
Synchronous Connection Complete event	1.2	Indicates to both the hosts that a new synchronous connection has been established.
Synchronous connection changed event	1.2	Indicates to the host that an existing synchronous connection has been reconfigured.
Accept Synchronous Connection Request command	1.2	Accepts an incoming request for a synchronous connection and informs the local LM about the acceptable parameter values for the synchronous connection.
Reject Synchronous Connection Request command	1.2	Declines an incoming request for a synchronous link.
Read Voice Setting command	1.1	Reads the values for the Voice_Setting configuration parameter, which controls all the various settings for the voice connections.
Write Voice Setting command	1.1	Writes the values for the Voice_Setting configuration parameter, which controls all the various settings for the voice connections.

11.3.10 Connection state

The connection state group of commands and events allows the configuration of a link, especially for low-power operation. See Table 80.

Table 80—Connection state

Name	Version	Summary description
Mode Change event	1.1	Indicates that the current mode has changed.
Max Slots Change event	1.1	Indicates a change in the maximum slots by the LM.
HOLD Mode command	1.1	Initiates HOLD mode.
SNIFF Mode command	1.1	Places a link specified by a connection handle into SNIFF mode.
Exit SNIFF mode command	1.1	Ends the SNIFF mode for a connection handle that is currently in SNIFF mode.
PARK State command	1.1	Places a link specified by a connection handle into the PARK state.
Exit PARK State command	1.1	Switches a link specified by a connection handle from PARK state back to active link.
Read Link Policy Settings command	1.1	Reads the Link_Policy configuration parameter for the specified connection handle. The link policy settings allow the host to specify which link modes the LM can use for the specified connection handle.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 80—Connection state *(continued)*

Name	Version	Summary description
Write Link Policy Settings command	1.1	Writes the Link_Policy configuration parameter for the specified connection handle. The link policy settings allow the host to specify which link modes the LM can use for the specified connection handle.
Read Default Link Policy Settings command	1.2	Reads the Default_Link_Policy configuration parameter for all new connections.
Write Default Link Policy Settings command	1.2	Writes the Default_Link_Policy configuration parameter for all new connections.

11.3.11 Piconet structure

The piconet structure group of commands and events allows the discovery and reconfiguration a piconet. See Table 81.

Table 81—Piconet structure

Name	Version	Summary description
Role Discovery command	1.1	Used for a device to determine which role the device is performing for a particular connection handle.
Switch Role command	1.1	Switches master and slave roles of the devices on either side of a connection.
Role Change event	1.1	Indicates that the current IEEE 802.15.1-2005 role related to the particular connection has been changed.

11.3.12 QoS

The QoS group of commands and events allows the configuration of links to allow for QoS parameters to be specified. See Table 82.

Table 82—QoS

Name	Version	Summary description
Flow Specification command	1.2	Specifies the flow parameters for the traffic carried over the ACL connection identified by the connection handle.
Flow Specification Complete event	1.2	Informs the host about the QoS for the ACL connection the controller is able to support.
QoS Setup command	1.1	Specifies QoS parameters for a connection handle.
QoS Setup Complete event	1.1	Indicates that QoS is setup.
QoS Violation event	1.1	Indicates the LM is unable to provide the current QoS requirement for the connection handle.
Flush command	1.1	Discards all data that are currently pending for transmission in the controller for the specified connection handle.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 82—QoS *(continued)*

Name	Version	Summary description
Flush Occurred event	1.1	Indicates that, for the specified connection handle, the data to be transmitted have been discarded.
Read Automatic Flush Timeout command	1.1	Reads the value for the Flush_Timeout configuration parameter for the specified connection handle. The Flush_Timeout parameter is used only for ACL connections.
Write Automatic Flush Timeout command	1.1	Writes the value for the Flush_Timeout configuration parameter for the specified connection handle. The Flush_Timeout parameter is used only for ACL connections.
Read Failed Contact Counter command	1.1	Reads the value for the Failed_Contact_Counter configuration parameter for a particular connection to another device.
Reset Failed Contact Counter command	1.1	Resets the value for the Failed_Contact_Counter configuration parameter for a particular connection to another device.
Read Num Broadcast Retransmissions command	1.1	Reads the parameter value for the number of broadcast retransmissions for the device.
Write Num Broadcast Retransmissions command	1.1	Writes the parameter value for the number of broadcast retransmissions for the device.

11.3.13 Physical links

The physical links group of commands and events allows configuration of the physical link. See Table 83.

Table 83—Physical links

Name	Version	Summary description
Read Link Supervision Timeout command	1.1	Reads the value for the Link_Supervision_Timeout configuration parameter for the device. This parameter is used by the device to determine link loss.
Write Link Supervision Timeout command	1.1	Writes the value for the Link_Supervision_Timeout configuration parameter for the device. This parameter is used by the device to determine link loss.
Read AFH Channel Assessment Mode command	1.2	Reads the value for the AFH_Channel_Classification_Mode configuration parameter. This value is used to enable or disable the controller's channel assessment scheme.
Write AFH Channel Assessment Mode command	1.2	Writes the value for the AFH_Channel_Classification_Mode configuration parameter. This value is used to enable or disable the controller's channel assessment scheme.
Set AFH Host Channel Classification command	1.2	Allows the host to specify a channel classification based on its local information.
Change Connection Packet Type command	1.1	Changes which packet types can be used for a connection that is currently established.
Connection Packet Type Changed event	1.1	Indicates the completion of the process of when the LM changes the packet type mask used for the specified connection handle.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.3.14 Host flow control

The host flow control group of commands and events allows flow control to be used toward the host. See Table 84.

Table 84—Controller flow control

Name	Version	Summary description
Host Buffer Size command	1.1	Used by the host to notify the controller about its buffer sizes for ACL and synchronous data. The controller will segment the data to be transmitted from the controller to the host so that data contained in HCI data packets will not exceed these sizes.
Set Event Mask command	1.1	Controls which events are generated by the HCI for the host.
Set Event Filter command	1.1	Used by the host to specify different event filters. The host may issue this command multiple times to request various conditions for the same type of event filter and for different types of event filters.
Set Controller To Host Flow Control command	1.1	Used by the host to turn flow control on or off in the direction from the controller to the host.
Host Number Of Completed Packets command	1.1	Used by the host to indicate to the controller when the host is ready to receive more HCI packets for any connection handle.
Data Buffer Overflow event	1.1	Indicates that the controller's data buffers have overflowed because the host has sent more packets than allowed.
Read Synchronous Flow Control Enable command	1.1	Provides the ability to read the "synchronous flow control enable" setting. The host uses this setting to find out whether the controller is currently configured to send Number Of Completed Packets events for synchronous connection handles.
Write Synchronous Flow Control Enable command	1.1	Provides the ability to write the "synchronous flow control enable" setting. By using this setting, the host can decide if the controller will send Number Of Completed Packets events for synchronous connection handles.

11.3.15 Link information

The link information group of commands and events allows information about a link to be read. See Table 85.

Table 85—Link information

Name	Version	Summary description
Read LMP Handle command	1.2	Reads the current LMP handle associated with the connection handle.
Read Transmit Power Level command	1.1	Reads the values for the Transmit_Power_Level parameter for the specified connection handle.
Read link quality command	1.1	Reads the value for the link quality for the specified connection handle.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 85—Link information *(continued)*

Name	Version	Summary description
Read RSSI command	1.1	Reads the value for the RSSI for a connection handle to another device.
Read Clock Offset command	1.1	Allows the host to read the clock offset of remote devices.
Read Clock Offset Complete event	1.1	Indicates the completion of the process of when the LM obtains the clock offset information.
Read Clock command	1.2	Reads an estimate of the master's clock or the CLKN.
Read AFH Channel Map command	1.2	Reads the current state of the channel map for a connection.

11.3.16 Authentication and encryption

The authentication and encryption group of commands and events allows authentication of a remote device and then encryption of the link to one or more remote devices. See Table 86.

Table 86—Authentication and encryption

Name	Version	Summary description
Read Authentication Enable command	1.1	Reads the value for the Authentication_Enable parameter, which controls whether the device will require authentication for each connection with other devices.
Write Authentication Enable command	1,1	Writes the value for the Authentication_Enable parameter, which controls whether the device will require authentication for each connection with other devices.
Read Encryption Mode command	1.1	Reads the value for the Encryption_Mode parameter, which controls whether the device will require encryption for each connection with other devices.
Write Encryption Mode command	1.1	Writes the value for the Encryption_Mode parameter, which controls whether the device will require encryption for each connection with other devices.
Link Key Request event	1.1	Indicates that a link key is required for the connection with the device specified in BD_ADDR.
Link Key Request Reply command	1.1	Replies to a Link Key Request event from the controller and specifies the link key stored on the host to be used as the link key for the connection with the other device specified by BD_ADDR.
Link Key Request Negative Reply command	1.1	Replies to a Link Key Request event from the controller if the host does not have a stored link key for the connection with the other device specified by BD_ADDR.
PIN Code Request event	1.1	Indicates that a PIN code is required to create a new link key for a connection.
PIN Code Request Reply command	1.1	Replies to a PIN Code Request event from the controller and specifies the PIN code to use for a connection.
PIN Code Request Negative Reply command	1.1	Replies to a PIN Code Request event from the controller when the host cannot specify a PIN code to use for a connection.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 86—Authentication and encryption *(continued)*

Name	Version	Summary description
Link Key Notification event	1.1	Indicates to the host that a new link key has been created for the connection with the device specified in BD_ADDR.
Authentication Requested command	1.1	Establishes authentication between the two devices associated with the specified connection handle.
Authentication Complete event	1.1	Occurs when authentication has been completed for the specified connection.
Set Connection Encryption command	1.1	Enables and disables the link-level encryption.
Encryption Change event	1.1	Indicates that the change in the encryption has been completed for the connection handle specified by the connection handle event parameter.
Change Connection Link Key command	1.1	Forces both devices of a connection associated to the connection handle to generate a new link key.
Change Connection Link Key Complete event	1.1	Indicates that the change in the link key for the connection handle specified by the connection handle event parameter had been completed.
Master Link Key command	1.1	Force both devices of a connection associated to the connection handle to use the temporary link key of the master device or the regular link keys.
Master Link Key Complete event	1.1	Indicates that the change in the temporary link key or in the semi-permanent link keys on the master side has been completed.
Read PIN Type command	1.1	Used for the host to read the value that is specified to indicate whether the host supports variable PIN or only fixed PINs.
Write PIN Type command	1.1	Used for the host to specify whether the host supports variable PIN or only fixed PINs.
Read Stored Link Key command	1.1	Provides the ability to read one or more link keys stored in the controller.
Return Link Keys event	1.1	Returns stored link keys after a Read Stored Link Key command is used.
Write Stored Link Key command	1.1	Provides the ability to write one or more link keys to be stored in the controller.
Delete Stored Link Key command	1.1	Provides the ability to remove one or more of the link keys stored in the controller.
Create New Unit Key command	1.1	Creates a new unit key.

11.3.17 Testing

The testing group of commands and events allows a device to be placed into a special testing mode to allow for testing to be performed. See Table 87.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 87—Testing

Name	Version	Summary description
Read Loopback Mode command	1.1	Reads the value for the setting of the controller's loopback mode. The setting of the loopback mode will determine the path of information.
Write Loopback Mode command	1.1	Writes the value for the setting of the controller's loopback mode. The setting of the loopback mode will determine the path of information.
Loopback Command event	1.1	Loops back all commands that the host sends to the controller with some exceptions.
Enable Device Under Test Mode command	1.1	Allows the local module to enter test mode via LMP test commands. The host issues this command when it wants the local device to be the DUT for testing purposes.

11.3.18 Alphabetical list of commands and events

See Table 88 for an alphabetical list of commands and events.

Table 88—Alphabetical list of commands and events

Commands/Events	Group
Accept Connection Request command	Connection setup
Authentication Complete event	Authentication and encryption
Authentication Requested command	Authentication and encryption
Change Connection Link Key command	Authentication and encryption
Change Connection Link Key Complete event	Authentication and encryption
Change Connection Packet Type command	Physical links
Command Complete event	Generic events
Command Status event	Generic events
Connection Complete event	Connection setup
Connection Packet Type Changed event	Physical links
Connection Request event	Connection setup
Create Connection Cancel command	Connection setup
Create Connection command	Connection setup
Create New Unit Key command	Authentication and encryption
Data Buffer Overflow event	Host flow control
Delete Stored Link Key command	Authentication and encryption
Disconnect command	Connection setup
Disconnection Complete event	Connection setup

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 88—Alphabetical list of commands and events *(continued)*

Commands/Events	Group
Enable Device Under Test Mode command	Testing
Encryption Change event	Authentication and encryption
Exit PARK State command	Connection state
Exit Periodic Inquiry Mode command	Device discovery
Exit SNIFF mode command	Connection state
Flow Specification command	QoS
Flow Specification Complete event	QoS
Flush command	QoS
Flush Occurred event	QoS
Hardware Error eventt	Generic events
HOLD Mode command	Connection state
Host Buffer Size command	Host flow control
Host Number Of Completed Packets command	Host flow control
Inquiry Cancel command	Device discovery
Inquiry command	Device discovery
Inquiry Complete event	Device discovery
Inquiry Result event	Device discovery
Inquiry Result With RSSI event	Device discovery
Link Key Notification event	Authentication and encryption
Link Key Request event	Authentication and encryption
Link Key Request Negative Reply command	Authentication and encryption
Link Key Request Reply command	Authentication and encryption
Loopback Command event	Testing
Master Link Key command	Authentication and encryption
Master Link Key Complete event	Authentication and encryption
Max Slots Change event	Connection state
Mode Change event	Connection state
Number Of Completed Packets event	Controller flow control
Page Scan Repetition Mode Change event	Connection setup
PARK State command	Connection state
Periodic Inquiry Mode command	Device discovery
PIN Code Request event	Authentication and encryption
PIN Code Request Negative Reply command	Authentication and encryption

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 88—Alphabetical list of commands and events *(continued)*

Commands/Events	Group
PIN Code Request Reply command	Authentication and encryption
QoS Setup command	QoS
QoS Setup Complete event	QoS
QoS Violation event	QoS
Read AFH Channel Assessment Mode command	Physical links
Read AFH Channel Map command	Link information
Read Authentication Enable command	Authentication and encryption
Read Automatic Flush Timeout command	QoS
Read BD_ADDR command	Controller information
Read Buffer Size command	Controller flow control
Read Class Of Device command	Controller information
Read Clock command	Link information
Read Clock Offset command	Link information
Read Clock Offset Complete event	Link information
Read Connection Accept Timeout command	Connection setup
Read Current IAC LAP command	Controller information
Read Default Link Policy Settings command	Connection state
Read Encryption Mode command	Authentication and encryption
Read Failed Contact Counter command	QoS
Read HOLD Mode Activity command	Connection state
Read Inquiry Mode command	Device discovery
Read Inquiry Scan Activity command	Device discovery
Read Inquiry Scan Type command	Device discovery
Read Link Policy Settings command	Connection state
Read link quality command	Link information
Read Link Supervision Timeout command	Physical links
Read LMP Handle command	Link information
Read Local Extended Features command	Controller information
Read Local Name command	Controller configuration
Read Local Supported Commands command	Controller information
Read Local Supported Features command	Controller information
Read Local Version Information command	Controller information
Read Loopback Mode command	Testing

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 88—Alphabetical list of commands and events *(continued)*

Commands/Events	Group
Read Num Broadcast Retransmissions command	QoS
Read Number Of Supported IAC command	Controller information
Read Page Scan Activity command	Connection setup
Read Page Scan Period Mode command	Connection setup
Read Page Scan Type command	Connection setup
Read Page Timeout command	Connection setup
Read PIN Type command	Authentication and encryption
Read Remote Extended Features command	Remote information
Read Remote Extended Features Complete event	Remote information
Read Remote Supported Features command	Remote information
Read Remote Supported Features Complete event	Remote information
Read Remote Version Information command	Remote information
Read Remote Version Information Complete event	Remote information
Read RSSI command	Link information
Read Scan Enable command	Controller information
Read Stored Link Key command	Authentication and encryption
Read Synchronous Flow Control Enable command	Host flow control
Read Transmit Power Level command	Link information
Read Voice Setting command	Synchronous connections
Reject Connection Request command	Connection setup
Remote Name Request Cancel command	Remote information
Remote Name Request command	Remote information
Remote Name Request Complete event	Remote information
Reset command	Device setup
Reset Failed Contact Counter command	QoS
Return Link Keys event	Authentication and encryption
Role Change event	Piconet structure
Role Discovery command	Piconet structure
Set AFH Host Channel Classification command	Physical links
Set Connection Encryption command	Authentication and encryption
Set Controller To Host Flow Control command	Host flow control
Set Event Filter command	Host flow control
Set Event Mask command	Host flow control

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 88—Alphabetical list of commands and events *(continued)*

Commands/Events	Group
Setup Synchronous Connection command	Synchronous connections
SNIFF Mode command	Connection state
Switch Role command	Piconet structure
Synchronous connection changed event	Synchronous connections
Synchronous Connection Complete event	Synchronous connections
Write AFH Channel Assessment Mode command	Physical links
Write Authentication Enable command	Authentication and encryption
Write Automatic Flush Timeout command	QoS
Write Class Of Device command	Controller information
Write Connection Accept Timeout command	Connection setup
Write Current IAC LAP command	Controller information
Write Default Link Policy Settings command	Connection state
Write Encryption Mode command	Authentication and encryption
Write HOLD Mode Activity command	Connection state
Write Inquiry Mode command	Device discovery
Write Inquiry Scan Activity command	Device discovery
Write Inquiry Scan Type command	Device discovery
Write Link Policy Settings command	Connection state
Write Link Supervision Timeout command	Physical links
Write Local Name command	Controller information
Write Loopback Mode command	Testing
Write Num Broadcast Retransmissions command	QoS
Write Page Scan Activity command	Connection setup
Write Page Scan Period Mode command	Connection setup
Write Page Scan Type command	Connection setup
Write Page Timeout command	Connection setup
Write PIN Type command	Authentication and encryption
Write Scan Enable command	Controller information
Write Stored Link Key command	Authentication and encryption
Write Synchronous Flow Control Enable command	Host flow control
Write Voice Setting command	Synchronous connections

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.4 HCI flow control

Flow control shall be used in the direction from the host to the controller to avoid overflowing the controller data buffers with ACL data destined for a remote device (using a connection handle) that is not responding. The host manages the data buffers of the controller.

11.4.1 Host-to-controller data flow control

On initialization, the host shall issue the Read Buffer Size command. Two of the return parameters of this command determine the maximum size of HCI ACL and synchronous data packets (excluding header) sent from the host to the controller. There are also two additional return parameters that specify the total number of HCI ACL and synchronous data packets that the controller may have waiting for transmission in its buffers.

When there is at least one connection to another device or when in local loopback mode, the controller shall use the Number Of Completed Packets event to control the flow of data from the host. This event contains a list of connection handles and a corresponding number of HCI data packets that have been completed (transmitted, flushed, or looped back to the host) since the previous time the event was returned (or since the connection was established if the event has not been returned before for a particular connection handle).

Based on the information returned in this event and the return parameters of the Read Buffer Size command that specify the total number of HCI ACL and synchronous data packets that can be stored in the controller, the host decides for which connection handles HCI data packets should be sent.

Every time it has sent an HCI data packet, the host shall assume that the free buffer space for the corresponding link type (ACL, SCO, or eSCO) in the controller has decreased by one HCI data packet.

Each Number Of Completed Packets event received by the host provides information about how many HCI data packets have been completed (transmitted or flushed) for each connection handle since the previous Number Of Completed Packets event was sent to the host. It can then calculate the actual current buffer usage.

When the controller has completed one or more HCI data packet(s), it shall send a Number Of Completed Packets event to the host until it finally reports that all the pending HCI data packets have been completed. The frequency at which this event is sent is manufacturer specific.

The Number Of Completed Packets events will not report on synchronous connection handles if synchronous flow control is disabled. (See 11.7.3.38 and 11.7.3.39.)

For each individual connection handle, the data must be sent to the controller in HCI data packets in the order in which it was created in the host. The controller shall also transmit data on the air that are received from the host for a given connection handle in the same order as they are received from the host.

Data that are received on the air from another device shall, for the corresponding connection handle, be sent in HCI data packets to the host in the same order as they are received. This means the scheduling shall be decided separately for each connection handle. For each individual connection handle, the order of the data shall not be changed from the order in which the data have been created.

11.4.2 Controller-to-host data flow control

In some implementations, flow control may also be necessary in the direction from the controller to the host. The Set Controller To Host Flow Control command can be used to turn flow control on or off in that direction.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

On initialization, the host uses the Host Buffer Size command to notify the controller about the maximum size of HCI ACL and synchronous data packets sent from the controller to the host. The command also contains two additional command parameters to notify the controller about the total number of ACL and synchronous data packets that can be stored in the data buffers of the host.

The host uses the Host Number Of Completed Packets command in exactly the same way as the controller uses the Number Of Completed Packets event as was previously described in 11.4.1.

The Host Number Of Completed Packets command is a special command for which no command flow control is used and which can be sent anytime there is a connection or when in local loopback mode. The command also has no event after the command has completed. This makes it possible for the flow control to work in exactly the same way in both directions, and the flow of normal commands will not be disturbed.

11.4.3 Disconnection behavior

When the host receives a Disconnection Complete event, the host shall assume that all unacknowledged HCI data packets that have been sent to the controller for the returned connection handle have been flushed and that the corresponding data buffers have been freed. The controller does not have to notify the host about this in a Number Of Completed Packets event.

If flow control is also enabled in the direction from the controller to the host, the controller may, after it has sent a Disconnection Complete event, assume that the host will flush its data buffers for the specified connection handle when it receives the Disconnection Complete event. The host does not have to notify the controller about this in a Host Number Of Completed Packets command.

11.4.4 Command flow control

On initial power-on, and after a reset, the host shall send a maximum of one outstanding HCI command packet until a Command Complete or Command Status event has been received.

The Command Complete and Command Status events contain a parameter called Num_HCI_Command_Packets, which indicates the number of HCI command packets the host is currently allowed to send to the controller. The controller may buffer one or more HCI command packets, but the controller shall start performing the commands in the order in which they are received. The controller may start performing a command before it completes previous commands. Therefore, the commands do not always complete in the order they are started.

To indicate to the host that the controller is ready to receive HCI command packets, the controller may generate a Command Complete event with the command opcode 0x0000, and the Num_HCI_Command_Packets event parameter set to 1 or more. Command opcode 0x0000 is a NOP (no operation) and may be used to change the number of outstanding HCI command packets that the host can send. The controller may generate a Command Complete event with the Num_HCI_Command_Packets event parameter set to zero to inform host it must stop sending commands.

For most commands, a Command Complete event shall be sent to the host when the controller completes the command. Some commands are executed in the background and do not return a Command Complete event when they have been completed. Instead, the controller shall send a Command Status event back to the host when it has begun to execute the command. When the actions associated with the command have finished, an event that is associated with the command shall be sent by the controller to the host.

If the command does not begin to execute, the Command Status event shall be returned with the appropriate error code in the Status parameter, and the event associated with the sent command shall not be returned. An example of such an instance is in the case where a parameter error or the command is currently not allowed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.4.5 Command error handling

If an error occurs for a command for which a Command Complete event is returned, the Return Parameters field may not contain all the return parameters specified for the command. The Status parameter, which explains the error reason and which is the first return parameter, shall always be returned. If there is a Connection_Handle parameter or a BD_ADDR parameter right after the Status parameter, this parameter shall also be returned so that the host can identify to which instance of a command the Command Complete event belongs. In this case, the Connection_Handle or BD_ADDR parameter shall have exactly the same value as that in the corresponding command parameter. It is implementation specific whether more parameters will be returned in case of an error.

The above also applies to commands that have associated command-specific completion events with a Status parameter in their completion event, with two exceptions. The exceptions are the Connection Complete and the Synchronous Connection Complete events. On failure, for these two events only, the second parameter, Connection_Handle, is not valid; and the third parameter, BD_ADDR, is valid for identification purposes. The validity of other parameters is likewise implementation specific for failed commands in this group.

The BD_ADDR return parameter of the Read BD_ADDR command is not used to identify to which instance of the Read BD_ADDR command the Command Complete event belongs. It is optional for the controller to return this parameter in case of an error.

11.5 HCI data formats

11.5.1 Introduction

The HCI provides a uniform command method of accessing the IEEE 802.15.1-2005 capabilities. The HCI link commands provide the host with the ability to control the link layer connections to other devices. These commands typically involve the LM to exchange LMP commands with remote devices. For details, see Clause 9.

The HCI policy commands are used to affect the behavior of the local and remote LM. These policy commands provide the host with methods of influencing how the LM manages the piconet. The controller-BB, informational, and status commands provide the host access to various registers in the controller.

HCI commands may take different amounts of time to be completed. Therefore, the results of commands will be reported back to the host in the form of an event. For example, for most HCI commands, the controller will generate the Command Complete event when a command is completed. This event contains the return parameters for the completed HCI command. For enabling the host to detect errors on the host controller transport layer, there needs to be a timeout between the transmission of the host's command and the reception of the controller's response (e.g., a Command Complete or Command Status event). Since the maximum response timeout is strongly dependent on the host controller transport layer used, it is recommended to use a default value of 1 s for this timer. This amount of time is also dependent on the number of commands unprocessed in the command queue.

11.5.2 Data and parameter formats

— All values are in binary and hexadecimal little-endian formats unless otherwise noted.
— In addition, all parameters that can have negative values must use twos complement when specifying values.
— Arrayed parameters are specified using the following notation: ParameterA[i]. If more than one set of arrayed parameters are specified (e.g., ParameterA[i], ParameterB[i]), then the order of the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

parameters are as follows: ParameterA[0], ParameterB[0], ParameterA[1], ParameterB[1], ParameterA[2], ParameterB[2], … ParameterA[n], ParameterB[n].

— Unless noted otherwise, all parameter values are sent and received in little-endian format [i.e., for multioctet parameters the rightmost (least signification octet) is transmitted first].
— All command and event parameters that are not arrayed and all elements in an arrayed parameter have fixed sizes (an integer number of octets). The parameters and the size of each not-arrayed parameter (or of each element in an arrayed parameter) contained in a command or an event is specified for each command or event. The number of elements in an arrayed parameter is not fixed.
— Where bit strings are specified, the low-order bit is the right-hand bit, e.g., 0 is the low-order bit in 10.

11.5.3 Connection handles

Connection handles are used to identify logical channels between the host and controller. Connection handles are assigned by the controller when a new logical link is created, using the Connection Complete or Synchronous Connection Complete event. Broadcast connection handles are handled differently and are described below in this subclause.

The first time the host sends an HCI data packet with Broadcast flag set to 01b (active broadcast) or 10b (piconet broadcast) after a power-on or a reset, the value of the Connection_Handle parameter must be a value that is not currently assigned by the host controller. The host must use different connection handles for active broadcast and piconet broadcast.

The host controller must then continue to use the same connection handles for each type of broadcast until a reset is made.

The host controller must not send a Connection Complete event containing a new connection handle that it knows is used for broadcast.

In some situations, it may happen that the host controller sends a Connection Complete event before having interpreted a broadcast packet received from the host and that the connection handles of both Connection Complete event and HCI data packet are the same. This conflict has to be avoided as follows:

— If a Connection Complete event is received containing one of the connection handles used for broadcast, the host has to wait before sending any packets for the new connection until it receives a Number Of Completed Packets event indicating that there are no pending broadcast packets belonging to the connection handle. In addition, the host must change the connection handle used for the corresponding type of broadcast to a connection handle that is currently not assigned by the host controller.
— This connection handle must then be used for all the following broadcasts of that type until a reset is performed or the same conflict situation happens again. However, this will occur very rarely.

The host controller must, in the conflict case in the previous paragraph, be able to distinguish between the broadcast message sent by the host and the new connection made (this could be even a new synchronous link) even though the connection handles are the same.

For an HCI data packet sent from the host controller to the host where the Broadcast_Flag is 01 or 10, the Connection_Handle parameter should contain the connection handle for the ACL connection to the master that sent the broadcast.

NOTE—Connection handles used for broadcast do not identify an ACL point-to-point connection, so they must not be used in any command having a Connection_Handle parameter and they will not be returned in any event having a Connection_Handle parameter except the Number Of Completed Packets event.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.5.4 Exchange of HCI-specific information

The host controller transport layer provides transparent exchange of HCI-specific information. These transporting mechanisms provide the ability for the host to send HCI commands, ACL data, and synchronous data to the controller. These transport mechanisms also provide the ability for the host to receive HCI events, ACL data, and synchronous data from the controller. Since the host controller transport layer provides transparent exchange of HCI-specific information, the HCI specification specifies the format of the commands, events, and data exchange between the host and the controller. The HCI packet formats are specified in 11.5.4.1 through 11.5.4.4.

11.5.4.1 HCI command packet

The HCI command packet is used to send commands to the controller from the host. The format of the HCI command packet is shown in Figure 93, and the definition of each field is explained in Table 89.

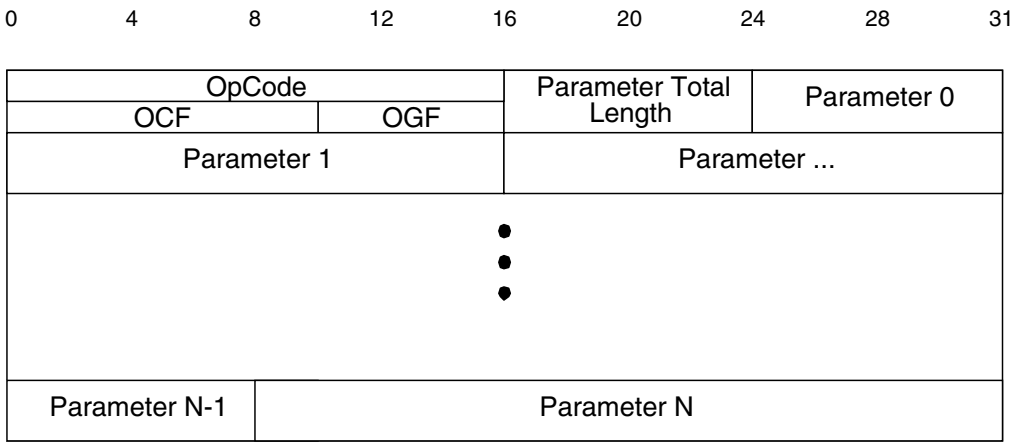


Figure 93—HCI command packet

The controller must be able to accept HCI command packets with up to 255 bytes of data excluding the HCI command packet header.

Each command is assigned a 2-byte opcode used to uniquely identify different types of commands. The Opcode parameter is divided into two fields: Opcode Group field (OGF) and Opcode Command field (OCF). The OGF occupies the upper 6 bits of the opcode, while the OCF occupies the remaining 10 bits. The OGF of 0x3F is reserved for vendor-specific debug commands. The OGF of 0x3E is reserved for Bluetooth logo testing. The organization of the opcodes allows additional information to be inferred without fully decoding the entire opcode.

The OGF composed of all ones has been reserved for vendor-specific debug commands. These commands are vendor-specific and are used during manufacturing, for a possible method for updating firmware, and for debugging.

The OGF composed of all zeros and an OCF of all zeros is the NOP command. This command opcode may be used in command flow control. (See 11.4.4.)

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 89—HCI command packet fields

Field	Size	Value	Parameter description
OpCode	2 octets	0xXXXX	OGF range (6 bits): 0x00–0x3F (0x3E reserved for Bluetooth logo testing and 0x3F reserved for vendor-specific debug commands) OCF range (10 bits): 0x0000–0x03FF
Parameter_ Total_Length	1 octet	0xXX	Lengths of all of the parameters contained in this packet measured in octets. (nb: total length of parameters, not number of parameters)
Parameter	*n* octetes	0xXX	Each command has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each command. Each parameter is an integer number of octets in size.

11.5.4.2 HCI ACL data packets

HCI ACL data packets are used to exchange data between the host and controller. The format of the HCI ACL data packet is shown in Figure 94. The definition for each of the fields in the data packets is explained in Table 90.

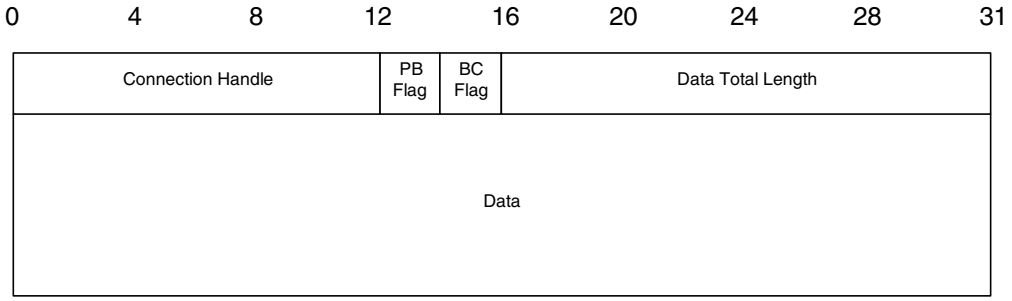


Figure 94—HCI ACL data packet

Table 90—HCI ACL data packet fields

Field	Size	Value	Parameter description
Connection _Handle	12 bits	0xXXX	Connection handle to be used for transmitting a data packet or segment. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Packet_ Boundary_ Flag	2 bits	00	Reserved for future use
		01	Continuing fragment packet of higher layer message
		10	First packet of higher layer message (i.e., start of an L2CAP packet)
		11	Reserved for future use

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 90—HCI ACL data packet fields *(continued)*

Field	Size	Value	Parameter description
Broadcast_ Flag (from host to controller)	2 bits	00	No broadcast. Only point-to-point.
		01	ASB: packet is sent to all active slaves (i.e., packet is usually not sent during park beacon slots), and it may be received by slaves in SNIFF mode or PARK state.
		10	PSB: packet is sent to all active slaves and all slaves in PARK state (i.e., packet is sent during park beacon slots if there are parked slaves), and it may be received by slaves in SNIFF mode.
		11	Reserved for future use
Broadcast_ Flag) (from con-troller to host)	2 bits	00	Point-to-point
		01	Packet received as a slave not in PARK state (either ASB or PSB)
		10	Packet received as a slave in PARK state (PSB)
		11	Reserved for future use.
Data_Total _Length	2 octets	0xXXXX	Length of data measured in octets.

The flag bits consist of the Packet_Boundary_Flag and Broadcast_Flag. The Packet_Boundary_Flag is located in bit 4 and bit 5, and the Broadcast_Flag is located in bit 6 and bit 7 in the second octet of the HCI ACL data packet.

ASB packets may be sent in park beacon slots.

NOTE—Slaves in SNIFF mode may or may not receive a broadcast packet depending on whether they happen to be listening at sniff slots, when the packet is sent.

11.5.4.3 HCI synchronous data packets

HCI synchronous (SCO and eSCO) data packets are used to exchange synchronous data between the host and controller. The format of the synchronous data packet is shown in Figure 95. The definition for each of the fields in the data packets is explained in Table 91.

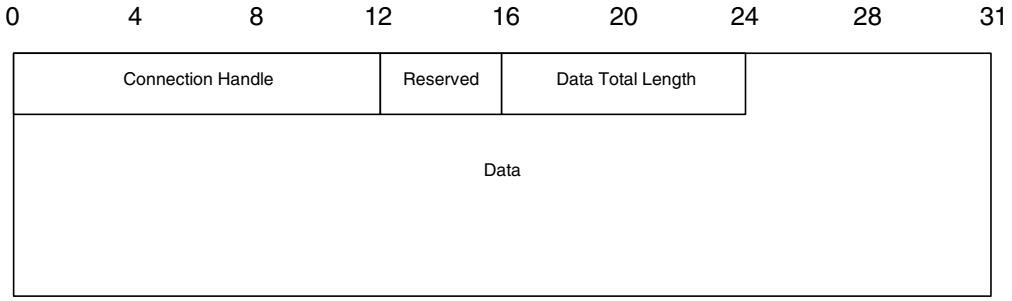


Figure 95—HCI synchronous data packet

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 91—HCI Synchronous data packet fields

Field	Size	Value	Parameter description
Connection _Handle	12 bits	0xXXX	Connection handle to be used for transmitting a synchronous data packet or segment. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Reserved	4 bits	XXXX	Reserved for future use
Data_Total _Length	1 octet	0xXX	Length of synchronous data measured in octets

11.5.4.4 HCI event packet

The HCI event packet is used by the controller to notify the host when events occur. The host must be able to accept HCI event packets with up to 255 octets of data excluding the HCI event packet header. The format of the HCI event packet is shown in Figure 96, and the definition of each field is explained in Table 92.

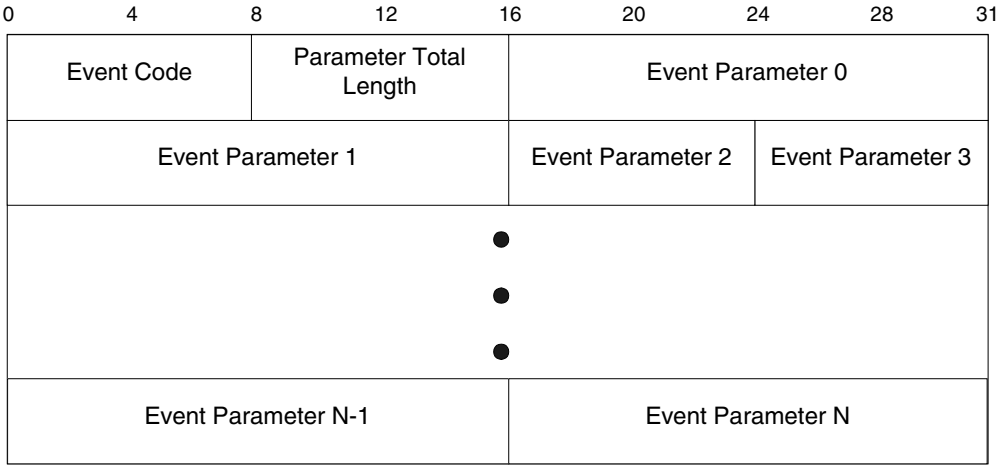


Figure 96—HCI event packet

Table 92—HCI event packet fields

Field	Size	Value	Parameter description
Event_ Code	1 octet	0xXX	Each event is assigned a 1-octet event code used to uniquely identify different types of events. Range: 0x00–0xFF (The event code 0xFF is reserved for the event code used for vendor-specific debug events. In addition, the event code 0xFE is also reserved for Bluetooth Logo Testing)
Parameter_ Total_ Length	1 octet	0xXX	Length of all of the parameters contained in this packet, measured in octets

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 92—HCI event packet fields *(continued)*

Field	Size	Value	Parameter description
Event_ Parameter 0–*N*	Size: parameter total length	0xXX	Each event has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each event. Each parameter is an integer number of octets in size.

11.6 HCI configuration parameters

11.6.1 Scan_Enable

The Scan_Enable parameter controls whether the device will periodically scan for page attempts and/or inquiry requests from other devices. If page scan is enabled, then the device will enter page scan mode based on the value of the Page_Scan_Interval and Page_Scan_Window parameters. If inquiry scan is enabled, then the device will enter inquiry scan mode based on the value of the Inquiry_Scan_Interval and Inquiry_Scan_Window parameters. See Table 93.

Table 93—Scan_Enable parameter values

Value	Parameter description
0x00	No scans enabled.
0x01	Inquiry scan enabled. Page scan always disabled.
0x02	Inquiry scan disabled. Page scan enabled.
0x03	Inquiry scan enabled. Page scan enabled.
0x04–0xFF	Reserved [0x04 limited page scan removed].

11.6.2 Inquiry_Scan_Interval

The Inquiry_Scan_Interval configuration parameter defines the amount of time between consecutive inquiry scans. This is defined as the time interval from when the controller started its last inquiry scan until it begins the next inquiry scan. See Table 94.

Table 94—Inquiry_Scan_Interval values

Value	Parameter description
N = 0xXXXX	Size: 2 octets Range: 0x0012–0x1000; only even values are valid Default: 0x1000 Mandatory Range: 0x0012–0x1000 Time = *N* * 0.625 ms Time Range: 11.25 to 2560 ms Time Default: 2.56 s

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.6.3 Inquiry_Scan_Window

The Inquiry_Scan_Window configuration parameter defines the amount of time for the duration of the inquiry scan. The inquiry scan window can be only less than or equal to the inquiry scan interval. See Table 95.

Table 95—Inquiry_Scan_Window values

Value	Parameter description
N = 0xXXXX	Size: 2 octets Range: 0x0011–0x1000 Default: 0x0012 Mandatory Range: 0x0011 to inquiry scan interval Time = *N* * 0.625 ms Time Range: 10.625 ms to 2560 ms Time Default: 11.25 ms

11.6.4 Inquiry_Scan_Type

The Inquiry_Scan_Type configuration parameter indicates whether inquiry scanning will be done using non-interlaced scan or interlaced scan. Currently, one mandatory inquiry scan type and one optional inquiry scan type are defined. See Table 96. For details, see 8.8.4.1.

Table 96—Inquiry_Scan_Type values

Value	Parameter description
0x00	Mandatory: standard scan (default)
0x01	Optional: interlaced scan
0x02–0xFF	Reserved

11.6.5 Inquiry_Mode

The Inquiry_Mode configuration parameter indicates whether inquiry returns Inquiry Result events in the standard format or with RSSI. See Table 97.

Table 97—Inquiry_Mode values

Value	Parameter description
0x00	Standard Inquiry Result event format
0x01	Inquiry Result event format with RSSI
0x02–0xFF	Reserved

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.6.6 Page_Reply_Timeout

The Page_Reply_Timeout configuration parameter defines the maximum time the local LM will wait for a BB page response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the page at BB level, the connection attempt will be considered to have failed. See Table 98.

Table 98—Page_Reply_Timeout values

Value	Parameter description
N = 0xXXXX	Size: 2 octets Range: 0x0001–0xFFFF Default: 0x2000 Mandatory Range: 0x0016–0xFFFF Time = *N* * 0.625 ms Time Range: 0.625 ms to 40.9 s Time Default: 5.12 s

11.6.7 Connection_Accept_Timeout

The Connection_Accept_Timeout configuration parameter allows the IEEE 802.15.1-2005 hardware to automatically deny a connection request after a specified time period has occurred, and the new connection is not accepted. The parameter defines the time duration from when the controller sends a Connection Request event until the controller will automatically reject an incoming connection. See Table 99.

Table 99—Connection_Accept_Timeout values

Value	Parameter description
N = 0xXXXX	Size: 2 octets Range: 0x0001–0xB540 Default: 0x1F40 Mandatory Range: 0x00A0–0xB540 Time = *N* * 0.625 ms Time Range: 0.625 ms to 29 s Time Default: 5 s

11.6.8 Page_Scan_Interval

The Page_Scan_Interval configuration parameter defines the amount of time between consecutive page scans. This time interval is defined from when the controller started its last page scan until it begins the next page scan. See Table 100.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 100—Page_Scan_Interval values

Value	Parameter description
N = 0xXXXX	Size: 2 octets Range: 0x0012–0x1000; only even values are valid Default: 0x0800 Mandatory Range: 0x0012–0x1000 Time = *N* * 0.625 ms Time Range: 11.25–2560 ms Time Default: 1.28 s

11.6.9 Page_Scan_Window

The Page_Scan_Window configuration parameter defines the amount of time for the duration of the page scan. The page scan window can be only less than or equal to the page scan interval. See Table 101.

Table 101—Page_Scan_Window values

Value	Parameter description
N = 0xXXXX	Size: 2 octets Range: 0x0011–0x1000 Default: 0x0012 Mandatory Range: 0x0011 to page scan interval Time = *N* * 0.625 ms Time Range: 10.625 ms to page scan interval Time Default: 11.25 ms

11.6.10 Page_Scan_Period_Mode

Every time an inquiry response message is sent, the device will start a timer (*T_mandatory_pscan*), the value of which is dependent on the page scan period mode. As long as this timer has not expired, the device will use the mandatory page scan mode for all following page scans. See Table 102.

Table 102—Page_Scan_Period_Mode values

Value	Parameter description
0x00	P0
0x01	P1
0x02	P2
0x03–0xFF	Reserved

The timer *T_mandatory_pscan* will be reset at each new inquiry response. For details, see 8.2.4.3.

11.6.11 Page_Scan_Type

The Page_Scan_Type parameter indicates whether inquiry scanning will be done using noninterlaced scan or interlaced scan. See Table 103. For details, see 8.8.3.1.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 103—Page_Scan_Type values

Value	Parameter description
0x00	Mandatory: standard scan (default)
0x01	Optional: interlaced scan
0x02–0xFF	Reserved

11.6.12 Voice_Setting

The Voice_Setting parameter controls all the various settings for voice connections. The input settings apply to all voice connections and cannot be set for individual voice connections. The Voice_Setting parameter controls the configuration for voice connections: input coding, air coding format, input data format, input sample size, and linear PCM parameter. The air coding format bits in the Voice_Setting command parameter specify which air coding format the local device requests. The air coding format bits do not specify which air coding format(s) the local device accepts when a remote device requests an air coding format. This is determined by the hardware capabilities of the local device. See Table 104.

Table 104—Voice_Setting values

Value	Parameter description
00XXXXXXXX	Input coding: linear
01XXXXXXXX	Input coding: μ-law input coding
10XXXXXXXX	Input coding: A-law input coding
11XXXXXXXX	Reserved for future use
XX00XXXXXX	Input data format: ones complement
XX01XXXXXX	Input data format: twos complement
XX10XXXXXX	Input data format: sign-magnitude
XX11XXXXXX	Input data format: unsigned
XXXX0XXXXX	Input sample size: 8-bit (only for linear PCM)
XXXX1XXXXX	Input sample size: 16-bit (only for linear PCM)
XXXXXnnnXX	Linear_PCM_Bit_Pos: # bit positions that MSB of sample is away from starting at MSB (only for liner PCM)
XXXXXXXX00	Air coding format: CVSD
XXXXXXXX01	Air coding format: μ-law
XXXXXXXX10	Air coding format: A-law
XXXXXXXX11	Air coding format: transparent data

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.6.13 PIN_Type

The PIN_Type configuration parameter determines whether the LM assumes that the host supports variable PIN codes or a fixed PIN code. The host controller uses the PIN-type information during pairing. See Table 105.

Table 105—PIN_Type values

Value	Parameter description
0x00	Variable PIN
0x01	Fixed PIN

11.6.14 Link_Key

The controller can store a limited number of link keys for other devices. Link keys are shared between two devices and are used for all security transactions between the two devices. A host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the controller when needed. A link key is associated with a BD_ADDR. See Table 106.

Table 106—Link_Key values

Value	Parameter description
0xXXXXXXXXXX XXXXXXXXXXX XXXXXXXXXX	Link key for an associated BD_ADDR

11.6.15 Authentication_Enable

The Authentication_Enable parameter controls if the local device requires authentication of the remote device at connection setup (between the Create Connection command or acceptance of an incoming ACL connection and the corresponding Connection Complete event). At connection setup, only the device(s) with the Authentication_Enable parameter enabled will try to authenticate the other device. See Table 107.

Changing this parameter does not affect existing connections.

Table 107—Authentication_Enable values

Value	Parameter description
0x00	Authentication not required
0x01	Authentication required for all connections
0x02–0xFF	Reserved

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.6.16 Encryption_Mode

The Encryption_Mode parameter controls if the local device requires encryption to the remote device at connection setup (between the Create Connection command or acceptance of an incoming ACL connection and the corresponding Connection Complete event). At connection setup, only devices with the Authentication_Enabled configuration parameter set to required and the Encryption_Mode configuration parameter set to required will try to encrypt the physical link to the other device.

Changing this parameter does not affect existing connections.

A temporary link key is used when both broadcast and point-to-point traffic are encrypted.

The host must not specify the Encryption_Mode parameter with more encryption capability than its local device currently supports, although the parameter is used to request the encryption capability to the remote device. Note that the host must not request the command with the Encryption_Mode parameter set to 0x01, when the local device does not support encryption. See Table 108.

Table 108—Encryption_Mode values

Value	Parameter description
0x00	Encryption not required
0x01	Encryption required for all connections
0x02–0xFF	Reserved

For encryption to be used, both devices must support and enable encryption.

In the Connection Complete event, the Encryption_Mode parameter will show whether encryption was successfully turned on. The remote device may not support encryption or may have set Encryption_Mode to 0x01 when the local device has not, so the encryption mode returned in the Connection Complete event may not equal the encryption mode set in the HCI Write Encryption Mode command.

11.6.17 Failed_Contact_Counter

The Failed_Contact_Counter parameter records the number of consecutive incidents in which either the slave or master did not respond after the flush timeout had expired, and the L2CAP packet that was currently being transmitted was automatically “flushed.” When this occurs, the failed contact counter is incremented by 1. See Table 109. The failed contact counter for a connection is reset to zero on the following conditions:

— When a new connection is established.
— When the failed contact counter is > zero and an L2CAP packet is acknowledged for that connection.
— When the Reset Failed Contact Counter command has been issued.

Table 109—Failed_Contact_Counter values

Value	Parameter description
0xXXXX	Number of consecutive failed contacts for a connection corresponding to the connection handle.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.6.18 HOLD_Mode_Activity

The Hold_Mode_Activity parameter value is used to determine what activities should be suspended when the device is in HOLD mode. After the hold period has expired, the device will return to the previous mode of operation. Multiple HOLD mode activities may be specified for the Hold_Mode_Activity parameter by performing a bitwise OR operation of the different activity types. If no activities are suspended, then all of the current periodic inquiry, inquiry scan, and page scan settings remain valid during the HOLD mode. If the Hold_Mode_Activity parameter is set to suspend page scan, suspend inquiry scan, and suspend periodic inquiries, then the device can enter a low-power state during the hold period, and all activities are suspended. Suspending multiple activities can be specified for the Hold_Mode_Activity parameter by performing a bitwise OR operation of the different activity types. The Hold_Mode_Activity parameter is valid only if all connections are in HOLD mode. See Table 110.

Table 110—HOLD_Mode_Activity values

Value	Parameter description
0x00	Maintain current power state
0x01	Suspend page scan
0x02	Suspend inquiry scan
0x04	Suspend periodic inquiries
0x08–0xFF	Reserved for future use

11.6.19 Link_Policy_Settings

The Link_Policy_Settings parameter determines the behavior of the local LM when it receives a request from a remote device or it determines itself to change the master-slave role or to enter PARK state, HOLD mode, or SNIFF mode. The local LM will automatically accept or reject such a request from the remote device and may even autonomously request itself, depending on the value of the Link_Policy_Settings parameter for the corresponding connection handle. When the value of the Link_Policy_Settings parameter is changed for a certain connection handle, the new value will be used only for requests from a remote device or from the local LM itself made after this command has been completed. By enabling each mode individually, the host can choose any combination needed to support various modes of operation. Multiple LM policies may be specified for the Link_Policy_Settings parameter by performing a bitwise OR operation of the different activity types. See Table 111.

Table 111—Link_Policy_Settings values

Value	Parameter description
0x0000	Disable all LM modes (default)
0x0001	Enable Role switch
0x0002	Enable HOLD mode
0x0004	Enable SNIFF mode
0x0008	Enable PARK state
0x0010–0x8000	Reserved for future use

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The local device may be forced into HOLD mode (regardless of whether the local device is master or slave) by the remote device regardless of the value of the Link_Policy_Settings parameter. The forcing of HOLD mode can, however, be done only once the connection has already been placed into HOLD mode through an LMP request (the link policy settings determine if requests from a remote device should be accepted or rejected). The forcing of HOLD mode can after that be done as long as the connection lasts regardless of the setting for HOLD mode in the Link_Policy_Settings parameter. This implies that if the implementation in the remote device is a "polite" implementation that does not force another device into HOLD mode via LMP PDUs, then the link policy settings will never be overruled.

11.6.20 Flush_Timeout

The Flush_Timeout configuration parameter is used for ACL connections only. The flush timeout is defined in 8.7.6.3. This parameter allows ACL packets to be automatically flushed without the host device issuing a Flush command. This provides support for isochronous data, such as audio. When the L2CAP packet that is currently being transmitted is automatically "flushed," the failed contact counter is incremented by one. See Table 112.

Table 112—Flush_Timeout values

Value	Parameter description
0	Timeout = ∞; no automatic flush
N = 0xXXXX	Size: 2 octets Range: 0x0001–0x07FF Mandatory Range: 0x0002–0x07FF Time = N * 0.625 ms Time Range: 0.625–1279.375 ms

11.6.21 Number_of_Broadcast_Retransmissions

Broadcast packets are not acknowledged and are unreliable. The Number_of_Broadcast_Retransmissions parameter N is used to increase the reliability of a broadcast message by retransmitting the broadcast message multiple times. This parameter defines the number of times the device will retransmit a broadcast data packet. This sets the value N_{BC} in the BB to one greater than the Number_of_Broadcast_Retransmissions value. (See 8.7.6.5.) This parameter should be adjusted as the link quality measurement changes. See Table 113.

Table 113—Number_of_Broadcast_Retransmissions values

Value	Parameter description
N = 0xXX	$N_{BC} = N + 1$ Range 0x00–0xFE

11.6.22 Link_Supervision_Timeout

The Link_Supervision_Timeout parameter is used by the master or slave device to monitor link loss. If, for any reason, no BB packets are received from that connection handle for a duration longer than the link supervision timeout, the connection is disconnected. The same timeout value is used for both synchronous and ACL connections for the device specified by the connection handle. See Table 114.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 114—Link_Supervision_Timeout values

Value	Parameter description
0x0000	No link supervision timeout.
N = 0xXXXX	Size: 2 octets Range: 0x0001–0xFFFF Default: 0x7D00 Mandatory Range: 0x0190–0xFFFF Time = *N* * 0.625 ms Time Range: 0.625 ms to 40.9 s Time Default: 20 s

Setting the Link_Supervision_Timeout parameter to "no link supervision timeout" (0x0000) will disable the link supervision timeout check for the specified connection handle. This makes it unnecessary for the master of the piconet to unpark and then park each device every ~40 s. By using the "no link supervision timeout" setting, the scalability of the PARK state is not limited.

11.6.23 Synchronous_Flow_Control_Enable

The Synchronous_Flow_Control_Enable confirguration parameter allows the host to decide if the controller will send Number Of Completed Packets events for synchronous connection handles. This setting allows the host to enable and disable synchronous flow control. See Table 115.

Table 115—Synchronous_Flow_Control_Enable values

Value	Parameter description
0x00	Synchronous flow control is disabled. No Number Of Completed Packets events will be sent from the controller for synchronous connection handles.
0x01	Synchronous flow control is enabled. Number Of Completed Packets events will be sent from the controller for synchronous connection handles.

11.6.24 Local_Name

The user-friendly local name provides the user the ability to distinguish one device from another. The Local_Name configuration parameter is a UTF-8 encoded string with up to 248 octets in length. The Local_Name configuration parameter will be null terminated (0x00) if the UTF-8 encoded string is less than 248 octets. See Table 116.

The Local_Name configuration parameter is a string parameter. Endianess does not, therefore, apply to the Local_Name configuration parameter. The first octet of the name is received first.

Table 116—Local_Name values

Value	Parameter description
	A UTF-8 encoded user-friendly descriptive name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.6.25 Class_of_Device

The Class_of_Device parameter is used to indicate the capabilities of the local device to other devices. See Table 117.

Table 117—Class_of_Device values

Value	Parameter description
0xXXXXXX	Class of the device

11.6.26 Supported_Commands

The Supported_Commands configuration parameter lists which HCI commands the local controller supports. It is implied that if a command is listed as supported, the feature underlying that command is also supported.

The Supported_Commands is a 64-octet bit field. If a bit is set to 1, then this command is supported. Table 118.

Table 118—Supported_Commands values

Octet	Bit	Command supported
0	0	Inquiry
	1	Inquiry Cancel
	2	Periodic Inquiry Mode
	3	Exit Periodic Inquiry Mode
	4	Create Connection
	5	Disconnect
	6	Add SCO Connection (deprecated)
	7	Create Connection Cancel
1	0	Accept Connection Request
	1	Reject Connection Request
	2	Link Key Request Reply
	3	Link Key Request Negative Reply
	4	PIN Code Request Reply
	5	PIN Code Request Negative Reply
	6	Change Connection Packet Type
	7	Authentication Requested

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 118—Supported_Commands values *(continued)*

Octet	Bit	Command supported
2	0	Set Connection Encryption
	1	Change Connection Link Key
	2	Master Link Key
	3	Remote Name Request
	4	Remote Name Request Cancel
	5	Read Remote Supported Features
	6	Read Remote Extended Features
	7	Read Remote Version Information
3	0	Read Clock Offset
	1	Read LMP Handle
	2	Reserved
	3	Reserved
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved
4	0	Reserved
	1	HOLD Mode
	2	SNIFF Mode
	3	Exit SNIFF Mode
	4	PARK State
	5	Exit PARK State
	6	QoS Setup
	7	Role Discovery
5	0	Switch Role
	1	Read Link Policy Settings
	2	Write Link Policy Settings
	3	Read Default Link Policy Settings
	4	Write Default Link Policy Settings
	5	Flow Specification
	6	Set Event Mask
	7	Reset

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 118—Supported_Commands values *(continued)*

Octet	Bit	Command supported
6	0	Set Event Filter
	1	Flush
	2	Read PIN Type
	3	Write PIN Type
	4	Create New Unit Key
	5	Read Stored Link Key
	6	Write Stored Link Key
	7	Delete Stored Link Key
7	0	Write Local Name
	1	Read Local Name
	2	Read Connection Accept Timeout
	3	Write Connection Accept Timeout
	4	Read Page Timeout
	5	Write Page Timeout
	6	Read Scan Enable
	7	Write Scan Enable
8	0	Read Page Scan Activity
	1	Write Page Scan Activity
	2	Read Inquiry Scan Activity
	3	Write Inquiry Scan Activity
	4	Read Authentication Enable
	5	Write Authentication Enable
	6	Read Encryption Mode
	7	Write Encryption Mode
9	0	Read Class Of Device
	1	Write Class Of Device
	2	Read Voice Setting
	3	Write Voice Setting
	4	Read Automatic Flush Timeout
	5	Write Automatic Flush Timeout
	6	Read Num Broadcast Retransmissions
	7	Write Num Broadcast Retransmissions

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 118—Supported_Commands values *(continued)*

Octet	Bit	Command supported
10	0	Read HOLD Mode Activity
	1	Write HOLD Mode Activity
	2	Read Transmit Power Level
	3	Read Synchronous Flow Control Enable
	4	Write Synchronous Flow Control Enable
	5	Set Controller To Host Flow Control
	6	Host Buffer Size
	7	Host Number Of Completed Packets
11	0	Read Link Supervision Timeout
	1	Write Link Supervision Timeout
	2	Read Number of Supported IAC
	3	Read Current IAC LAP
	4	Write Current IAC LAP
	5	Read Page Scan Period Mode
	6	Write Page Scan Period Mode
	7	Read Page Scan Mode (deprecated)
12	0	Write Page Scan Mode (deprecated)
	1	Set AFH Host Channel Classification
	2	Reserved
	3	Reserved
	4	Read Inquiry Scan Type
	5	Write Inquiry Scan Type
	6	Read Inquiry Mode
	7	Write Inquiry Mode
13	0	Read Page Scan Type
	1	Write Page Scan Type
	2	Read AFH Channel Assessment Mode
	3	Write AFH Channel Assessment Mode
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 118—Supported_Commands values *(continued)*

Octet	Bit	Command supported
14	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Version Information
	4	Reserved
	5	Read Local Supported Features
	6	Read Local Extended Features
	7	Read Buffer Size
15	0	Read Country Code (deprecated)
	1	Read BD_ADDR
	2	Read Failed Contact Counter
	3	Reset Failed Contact Counter
	4	Read Link Quality
	5	Read RSSI
	6	Read AFH Channel Map
	7	Read Clock
16	0	Read Loopback Mode
	1	Write Loopback Mode
	2	Enable Device Under Test Mode
	3	Setup Synchronous Connection
	4	Accept Synchronous Connection
	5	Reject Synchronous Connection[
	6	Reserved
	7	Reserved

11.7 HCI commands and events

HCI commands and events are documented in the general format for each command:

— Clause heading of the command
— Paragraph(s) describing the function of the command
— Table containing the command format and parameters
— Table specifying allowable sizes and values of parameters (if applicable)
— Table specifying return values (if applicable)
— Paragraphs describing the events that may be generated by the command, unless they are masked away

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.1 Link control commands

The link control commands allow the controller to control connections to other devices. When the link control commands are used, the LM controls how the IEEE 802.15.1-2005 piconets and scatternets are established and maintained. These commands instruct the LM to create and modify link layer connections with IEEE 802.15.1-2005 remote devices, perform inquiries of other devices in range, and perform other LMP commands. For the link control commands, the OGF is defined as 0x01.

11.7.1.1 Inquiry command

This command will cause the device to enter inquiry mode. Inquiry mode is used to discover other nearby devices. The LAP input parameter contains the LAP from which the IAC shall be derived when the inquiry procedure is made. The Inquiry_Length parameter specifies the total duration of the inquiry mode; and, when this time expires, inquiry will be halted. The Num_Responses parameter specifies the number of responses that can be received before the inquiry is halted. A Command Status event is sent from the controller to the host when the Inquiry command has been started by the device. When the inquiry process is completed, the controller will send an Inquiry Complete event to the host indicating that the inquiry has finished. The event parameters of the Inquiry Complete event will have a summary of the result from the inquiry process, which reports the number of nearby devices that responded. When a device responds to the inquiry message, an Inquiry Result event will occur to notify the host of the discovery.

A device that responds during an inquiry or inquiry period should always be reported to the host in an Inquiry Result event if the device has not been reported earlier during the current inquiry or inquiry period and the device has not been filtered out using the Set Event Filter command. If the device has been reported earlier during the current inquiry or inquiry period, it may or may not be reported depending on the implementation (depending on if earlier results have been saved in the controller and in that case how many responses that have been saved). It is recommended that the controller try to report a particular device only once during an inquiry or inquiry period. See Table 119 and Table 120.

Table 119—Inquiry command

Command	OCF	Command parameters	Return parameters
HCI_Inquiry	0x0001	LAP, Inquiry_Length, Num_Responses	None

Table 120—Inquiry command values

Parameter	Size	Value	Parameter description
LAP	3 octets	0x9E8B00–0X9E8B3F	This is the LAP from which the IAC should be derived when the inquiry procedure is made (see Bluetooth Assigned Numbers [B1]).
Inquiry_ Length	1 octet	*N* = 0xXX	Maximum amount of time specified before the inquiry is halted. Size: 1 octet Range: 0x01–0x30 Time = *N* * 1.28 s Range: 1.28–61.44 s
Num_ Responses	1 octet	0x00	Unlimited number of responses.
		0xXX	Maximum number of responses from the inquiry before the inquiry is halted. Range: 0x01–0xFF

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A Command Status event is sent from the controller to the host when the controller has started the inquiry process. An Inquiry Result event will be created for each device that responds to the inquiry message. In addition, multiple devices that respond to the inquiry message may be combined into the same event. An Inquiry Complete event is generated when the inquiry process has completed. Note that no Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Inquiry Complete event will indicate that this command has been completed. No Inquiry Complete event will be generated for the canceled inquiry process.

11.7.1.2 Inquiry Cancel command

This command will cause the device to stop the current inquiry if the device is in inquiry mode. This command allows the host to interrupt the device and request the device to perform a different task. The command should be issued only after the Inquiry command has been issued, after a Command Status event has been received for the Inquiry command, and before the Inquiry Complete event occurs. See Table 121 and Table 122.

Table 121—Inquiry Cancel command

Command	OCF	Command parameters	Return parameters
HCI_Inquiry_Cancel	0x0002		Status

Table 122—Inquiry Cancel command return values

Return parameter	Size	Value	Parameter description
Status	1 octet	0x00	Inquiry Cancel command succeeded.
		0x01–0xFF	Inquiry Cancel command failed. See 10.3 for list of error codes.

When the Inquiry Cancel command has completed, a Command Complete event will be generated. No Inquiry Complete event will be generated for the canceled inquiry process.

11.7.1.3 Periodic Inquiry Mode command

The Periodic Inquiry Mode command is used to configure the device to enter the periodic inquiry mode that performs an automatic inquiry. Max_Period_Length and Min_Period_Length parameters define the time range between two consecutive inquiries, from the beginning of an inquiry until the start of the next inquiry. The controller will use this range to determine a new random time between two consecutive inquiries for each inquiry. The LAP input parameter contains the LAP from which the IAC shall be derived when the inquiry procedure is made. The Inquiry_Length parameter specifies the total duration of the inquiry mode; and, when time expires, inquiry will be halted. The Num_Responses parameter specifies the number of responses that can be received before the inquiry is halted. This command is completed when the inquiry process has been started by the device, and a Command Complete event is sent from the controller to the host. When each of the periodic inquiry processes are completed, the controller will send an Inquiry Complete event to the host indicating that the latest periodic inquiry process has finished. When a device responds to the inquiry message, an Inquiry Result event will occur to notify the host of the discovery.

Maximum period length > minimum period length > inquiry length.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A device that responds during an inquiry or inquiry period should always be reported to the host in an Inquiry Result event if the device has not been reported earlier during the current inquiry or inquiry period and the device has not been filtered out using the Set Event Filter command. If the device has been reported earlier during the current inquiry or inquiry period, it may or may not be reported depending on the implementation (depending on if earlier results have been saved in the controller and in that case how many responses that have been saved). It is recommended that the controller try to report a particular device only once during an inquiry or inquiry period. See Table 123, Table 124, and Table 125.

Table 123—Periodic Inquiry Mode command

Command	OCF	Command parameters	Return parameters
HCI_Periodic_Inquiry_Mode	0x0003	Max_Period_Length, Min_Period_Length, LAP, Inquiry_Length, Num_Responses	Status

Table 124—Periodic Inquiry Mode command paramters

Parameter	Size	Value	Parameter description
Max_ Period_ Length	2 octets	*N* = 0xXXXX	Maximum amount of time specified between consecutive inquiries. Range: 0x03–0xFFFF Time = *N* * 1.28 s Range: 3.84–83884.8 s 0.0–23.3 hr
Min_ Period_ Length	2 octets	*N* = 0xXXXX	Minimum amount of time specified between consecutive inquiries. Range: 0x02–0xFFFE Time = *N* * 1.28 s Range: 2.56–83883.52 s 0.0–23.3 hr
LAP	3 octets	0x9E8B00– 0X9E8B3F	This is the LAP from which the IAC should be derived when the inquiry procedure is made (see Bluetooth Assigned Numbers [B1]).
Inquiry_ Length	1 octet	*N* = 0xXX	Maximum amount of time specified before the inquiry is halted. Range: 0x01–0x30 Time = *N* * 1.28 s Range: 1.28–61.44 s
Num_ Responses	1 octet	0x00	Unlimited number of responses.
		0xXX	Maximum number of responses from the inquiry before the inquiry is halted. Range: 0x01–0xFF

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 125—Periodic Inquiry Mode command status return

Return parameter	Size	Value	Parameter description
Status	1 octet	0x00	Periodic Inquiry Mode command succeeded.
		0x01–0xFF	Periodic Inquiry Mode command failed. See 10.3 for list of error codes.

The periodic inquiry mode begins when the controller sends the Command Complete event for this command to the host. An Inquiry Result event will be created for each device that responds to the inquiry message. In addition, multiple devices that respond to the inquiry message may be combined into the same event. An Inquiry Complete event is generated when each of the periodic inquiry processes has completed. No Inquiry Complete event will be generated for the canceled inquiry process.

11.7.1.4 Exit Periodic Inquiry Mode command

The Exit Periodic Inquiry Mode command is used to end the periodic inquiry mode when the local device is in periodic inquiry mode. If the local device is currently in an inquiry process, the inquiry process will be stopped directly, and the controller will no longer perform periodic inquiries until the Periodic Inquiry Mode command is reissued. See Table 126 and Table 127.

Table 126—Exit Periodic Inquiry Mode command

Command	OCF	Command parameters	Return parameters
HCI_Exit_Periodic_Inquiry_Mode	0x0004	None	Status

Table 127—Exit_Periodic_Inquiry_Mode command return status

Return parameter	Size	Value	Parameter description
Status	1 octet	0x00	Exit Periodic Inquiry Mode command succeeded.
		0x01–0xFF	Exit Periodic Inquiry Mode command failed. See 10.3 for list of error codes.

A Command Complete event for this command will occur when the local device is no longer in periodic inquiry mode. No Inquiry Complete event will be generated for the canceled inquiry process.

11.7.1.5 Create Connection command

This command will cause the LM to create a connection to the device with the BD_ADDR specified by the command parameters. This command causes the local device to begin the page process to create a link-level connection. The LM will determine how the new ACL connection is established. This ACL connection is determined by the current state of the device, its piconet, and the state of the device to be connected. The Packet_Type command parameter specifies which packet types the LM shall use for the ACL connection. When sending HCI ACL data packets, the LM shall use only the packet type(s) specified by the

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Packet_Type command parameter or the always-allowed **DM1** packet type. Multiple packet types may be specified for the Packet_Type parameter by performing a bitwise OR operation of the different packet types. The LM may choose which packet type to be used from the list of acceptable packet types. The Page_Scan_Repetition_Mode parameter specifies the page scan repetition mode supported by the remote device with the BD_ADDR. This is the information that was acquired during the inquiry process. The Clock_Offset parameter is the difference between its own clock and the clock of the remote device with BD_ADDR. Only bits 2 through 16 of the difference are used, and they are mapped to this parameter as bits 0 through 14, respectively. A Clock Offset Valid flag, located in bit 15 of the Clock_Offset parameter, is used to indicate if the clock offset is valid or not. A connection handle for this connection is returned in the Connection Complete event (see the note right after this paragraph). The Allow_Role_Switch parameter specifies if the local device accepts or rejects the request of a master-slave role switch when the remote device requests it at the connection setup (in the Role parameter of the Accept Connection Request command) (before the local controller returns a Connection Complete event). See Table 128 and Table 129. For a definition of the different packet types, see Clause 8.

NOTE—The host should enable as many packet types as possible for the LM to perform efficiently. However, the host must not enable packet types that the local device does not support.

Table 128—Create_Connection command

Command	OCF	Command parameters	Return parameters
HCI_Create_Connection	0x0005	BD_ADDR, Packet_Type, Page_Scan_Repetition_Mode, Reserved, Clock_Offset, Allow_Role_Switch	None

Table 129—Create_Connection command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR of the device to be connected.
Packet_Type	2 octets	0x0001	Reserved for future use.
		0x0002	Reserved for future use.
		0x0004	Reserved for future use.
		0x0008[a]	**DM1**
		0x0010	**DH1**
		0x0020	Reserved for future use.
		0x0040	Reserved for future use.
		0x0080	Reserved for future use.
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	**DM3**

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 129—Create_Connection command parameters *(continued)*

Parameter	Size	Value	Parameter description
Packet_Type *(continued)*		0x0800	**DH3**
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	**DM5**
		0x8000	**DH5**
Page_Scan_ Repetition_ Mode	1 octet	0x00	R0
		0x01	R1
		0x02	R2
		0x03–0xFF	Reserved.
Reserved	1 octet	0x00	Reserved, must be set to 0x00. See 11.8.1.
Clock_Offset	2 octets	Bit 14–0	Bit 16–2 of CLK_{slave}–CLK_{master}
		Bit 15	Clock Offset Valid flag Invalid clock offset = 0 Valid clock offset = 1
Allow_Role_ Switch	1 octet	0x00	The local device will be a master and will not accept a role switch requested by the remote device at the connection setup.
		0x01	The local device may be a master or may become a slave after accepting a role switch requested by the remote device at the connection setup.
		0x02–0xFF	Reserved for future use.

[a]This bit will be interpreted as set to 1 by link controllers based on IEEE Std 802.15.1-2005 or newer.

When the controller receives the Create Connection command, the controller sends the Command Status event to the host. In addition, when the LM determines the connection is established, the controller, on both devices that form the connection, will send a Connection Complete event to each host. The Connection Complete event contains the connection handle if this command is successful.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

11.7.1.6 Disconnect command

The Disconnect command is used to terminate an existing connection. The Connection_Handle command parameter indicates which connection is to be disconnected. The Reason command parameter indicates the reason for ending the connection. The remote device will receive the Reason command parameter in the Disconnection Complete event. All synchronous connections on a physical link should be disconnected before the ACL connection on the same physical connection is disconnected. See Table 130 and Table 131.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 130—Disconnect command

Command	OCF	Command parameters	Return parameters
HCI_Disconnect	0x0006	Connection_Handle, Reason	None

Table 131—Disconnect command parameters

Parameter	Size	Value	Parameter description
Connection _Handle	2 octets	0xXXXX	Connection handle for the connection being disconnected. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Reason	1 octet	0x05, 0x13–0x15, 0x1A, 0x29	Error codes *authentication failure* (0x05), *remote device terminated connection* (0x13–0x15), *unsupported remote feature* (0x1A), and *pairing with unit key not supported* (0x29). See 10.3 for list of error codes.

When the controller receives the Disconnect command, it sends the Command Status event to the host. The Disconnection Complete event will occur at each host when the termination of the connection has completed and indicates that this command has been completed.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Disconnection Complete event will indicate that this command has been completed.

11.7.1.7 Create Connection Cancel command

This command is used to request cancellation of the ongoing connection creation process, which was started by a Create Connection command of the local device. See Table 132, Table 133, and Table 134.

Table 132—Create Connection Cancel command

Command	OCF	Command parameters	Return parameters
HCI_Create_Connection_ Cancel	0x0008	BD_ADDR	Status, BD_ADDR

Table 133—Create Connection Cancel command parameter

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which the Create Connection command was issued before and that is the subject of this cancellation request.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 134—Create_Connection_Cancel command return parameters

Return Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Create Connection Cancel command succeeded
		0x01–0xff	Create Connection Cancel command failed. See 10.3 for list of error codes.
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which the Create Connection command was issued before and is the subject of this cancellation request.

When the Create Connection Cancel command has completed, a Command Complete event shall be generated.

If the connection is already established by the BB, but the controller has not yet sent the Connection Complete event, then the local device shall detach the link and return a Command Complete event with the status *success*.

If the connection is already established and the Connection Complete event has been sent, then the controller shall return a Command Complete event with the error code *ACL connection already exists* (0x0B).

If the Create Connection Cancel command is sent to the controller without a preceding Create Connection command to the same device, the controller shall return a Command Complete event with the error code *unknown connection identifier* (0x02).

The error codes *unspecified error* (0x1F) and *command disallowed* (0x0C) may be used if the controller does not support this command.

The Connection Complete event for the corresponding Create Connection command shall always be sent. The Connection Complete event shall be sent after the Command Complete event for the Create Connection Cancel command. If the cancellation was successful, the Connection Complete event will be generated with the error code *unknown connection identifier* (0x02).

11.7.1.8 Accept Connection Request command

The Accept Connection Request command is used to accept a new incoming connection request. The Accept Connection Request command shall be issued only after a Connection Request event has occurred. The Connection Request event will return the BD_ADDR of the device that is requesting the connection. This command will cause the LM to create a connection to the device, with the BD_ADDR specified by the command parameters. The LM will determine how the new connection will be established. This will be determined by the current state of the device, its piconet, and the state of the device to be connected. The Role command parameter allows the host to specify if the LM shall request a role switch and become the master for this connection. This is a preference and not a requirement. If the role switch fails, then the connection will still be accepted, and the Role Discovery command will reflect the current role. See Table 135 and Table 136.

The LM may terminate the connection if it would be low on resources if the role switch fails. The decision to accept a connection must be completed before the connection accept timeout expires on the local IEEE 802.15.1-2005 module.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 135—Accept Connection Request command

Command	OCF	Command parameters	Return parameters
HCI_Accept_Connection_ Request	0x0009	BD_ADDR, Role	None

Table 136—Accept Connection Request command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXX XXXXXX X	BD_ADDR of the device to be connected.
Role	1 octet	0x00	Become the master for this connection. The LM will perform the role switch.
		0x01	Remain the slave for this connection. The LM will not perform the role switch.

When accepting synchronous connection request, the Role parameter is not used and will be ignored by the controller.

The Accept Connection Request command will cause the Command Status event to be sent from the controller when the controller begins setting up the connection. In addition, when the LM determines the connection is established, the local controller will send a Connection Complete event to its host, and the remote controller will send a Connection Complete event or a Synchronous Connection Complete event to the host. The Connection Complete event contains the connection handle if this command is successful.

NOTE—No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

11.7.1.9 Reject Connection Request command

The Reject Connection Request command is used to decline a new incoming connection request. The Reject Connection Request command shall be called only after a Connection Request event has occurred. The Connection Request event will return the BD_ADDR of the device that is requesting the connection. The Reason command parameter will be returned to the connecting device in the Status parameter of the Connection Complete event returned to the host of the connection device to indicate why the connection was declined. See Table 137 and Table 138.

Table 137—Reject Connection Request command

Command	OCF	Command parameters	Return parameters
HCI_Reject_Connection_ Request	0x000A	BD_ADDR, Reason	None

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 138—Reject Connection Request command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR of the device from which to reject the connection.
Reason	1 octet	0x0D–0x0F	See 10.3 for list of error codes and descriptions.

When the controller receives the Reject Connection Request command, the controller sends the Command Status event to the host. Then, the local controller will send a Connection Complete event to its host, and the remote controller will send a Connection Complete event or a Synchronous Connection Complete event to the host. The Status parameter of the Connection Complete event, which is sent to the host of the device attempting to make the connection, will contain the Reason command parameter from this command.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

11.7.1.10 Link Key Request Reply command

The Link Key Request Reply command is used to reply to a Link Key Request event from the controller and specifies the link key stored on the host to be used as the link key for the connection with the other device specified by BD_ADDR. The Link Key Request event will be generated when the controller needs a link key for a connection.

When the controller generates a Link Key Request event in order for the local LM to respond to the request from the remote LM (as a result of a Create Connection or Authentication Requested command from the remote host), the local host must respond with either a Link Key Request Reply or Link Key Request Negative Reply command before the remote LM detects LMP response timeout. (See Clause 9.) See Table 139, Table 140, and Table 141.

Table 139—Link Key Request Reply command

Command	OCF	Command parameters	Return parameters
HCI_Link_Key_Request_Reply	0x000B	BD_ADDR, Link_Key	Status, BD_ADDR

Table 140—Link Key Request Reply command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR of the device for which the link key is associated.
Link_Key	16 octets	0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Link key for the associated BD_ADDR.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 141—Link Key Request Reply command return parameters

Return parameter	Size	Value	Parameter description
Status	1 octet	0x00	Link Key Request Reply command succeeded.
		0x01–0xFF	Link Key Request Reply command failed. See 10.3 for error codes and description.
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which the Link Key Request Reply command has completed.

The Link Key Request Reply command will cause a Command Complete event to be generated.

11.7.1.11 Link Key Request Negative Reply command

The Link Key Request Negative Reply command is used to reply to a Link Key Request event from the controller if the host does not have a stored link key for the connection with the other device specified by BD_ADDR. The Link Key Request event will be generated when the controller needs a link key for a connection.

When the controller generates a Link Key Request event in order for the local LM to respond to the request from the remote LM (as a result of a Create Connection or Authentication Requested command from the remote host), the local host must respond with either a Link Key Request Reply or Link Key Request Negative Reply command before the remote LM detects LMP response timeout. (See Clause 9.) See Table 142, Table 143, and Table 144.

Table 142—Link Key Request Negative Reply command

Command	OCF	Command parameters	Return parameters
HCI_Link_Key_Request_ Negative_Reply	0x000C	BD_ADDR	Status, BD_ADDR

Table 143—Link Key Request Negative Reply command parameter

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR of the device for which the link key is associated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 144—Link Key Request Negative Reply command return parameters

Return parameter	Size	Value	Parameter description
Status	1 octet	0x00	Link Key Request Negative Reply command succeeded.
		0x01–0xFF	Link Key Request Negative Reply command failed. See 10.3 for list of error codes.
BD_ADDR	6 octets	0xXXXXXXXXXXX	BD_ADDR of the device for which the Link Key Request Negative Reply command has completed.

The Link Key Request Negative Reply command will cause a Command Complete event to be generated.

11.7.1.12 PIN Code Request Reply command

The PIN Code Request Reply command is used to reply to a PIN Code Request event from the controller and specifies the PIN code to use for a connection. The PIN Code Request event will be generated when a connection with remote initiating device has requested pairing.

When the controller generates a PIN Code Request event in order for the local LM to respond to the request from the remote LM (as a result of a Create Connection or Authentication Requested command from the remote host), the local host must respond with either a PIN Code Request Reply or PIN Code Request Negative Reply command before the remote LM detects LMP response timeout. (See Clause 9.) See Table 145, Table 146, and Table 147.

Table 145—PIN Code Request Reply command

Command	OCF	Command parameters	Return parameters
HCI_PIN_Code_Request_Reply	0x000D	BD_ADDR, PIN_Code_Length, PIN_Code	Status, BD_ADDR

Table 146—PIN Code Request Reply command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXX	BD_ADDR of the device for which the PIN code is associated.
PIN_Code_ Length	1 octet	0xXX	The PIN code length specifics the length, in octets, of the PIN code to be used. Range: 0x01–0x10
PIN_Code	16 octets	0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	PIN code for the device that is to be connected. The host should ensure that strong PIN codes are used. PIN codes can be up to a maximum of 128 bits. NOTE—The PIN_Code parameter is a string parameter. Endianess does not, therefore, apply to the PIN_Code parameter. The first octet of the PIN code should be transmitted first.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 147—PIN Code Request Reply command parameters returned

Returned parameter	Size	Value	Parameter description
Status	1 octet	0x00	PIN Code Request Reply command succeeded.
		0x01–0xFF	PIN Code Request Reply command failed. See 10.3 for list of error codes.
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which the PIN Code Request Reply command has completed.

The PIN Code Request Reply command will cause a Command Complete event to be generated.

11.7.1.13 PIN Code Request Negative Reply command

The PIN Code Request Negative Reply command is used to reply to a PIN Code Request event from the controller when the host cannot specify a PIN code to use for a connection. This command will cause the pair request with remote device to fail.

When the controller generates a PIN Code Request event in order for the local LM to respond to the request from the remote LM (as a result of a Create Connection or Authentication Requested command from the remote host), the local host must respond with either a PIN Code Request Reply or PIN Code Request Negative Reply command before the remote LM detects LMP response timeout. (See Clause 9.) See Table 148, Table 149, and Table 150.

Table 148—PIN Code Request Negative Reply command

Command	OCF	Command parameters	Return parameters
HCI_PIN_Code_ Request_Negative_Reply	0x000E	BD_ADDR	Status, BD_ADDR

Table 149—PIN Code Request Negative Reply command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXX	BD_ADDR of the device to which this command is responding.
Status	1 octet	0x00	PIN Code Request Negative Reply command succeeded.
		0x01–0xFF	PIN Code Request Negative Reply command failed. See 10.3 for list of error codes.

Table 150—PIN code request negative reply command returned parameters

Return parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which the PIN Code Request Negative Reply command has completed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The PIN Code Request Negative Reply command will cause a Command Complete event to be generated.

11.7.1.14 Change Connection Packet Type command

The Change Connection Packet Type command is used to change which packet types can be used for a connection that is currently established. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type command parameter specifies which packet types the LM can use for the connection. When sending HCI ACL data packets, the LM shall use only the packet type(s) specified by the Packet_Type command parameter or the always-allowed **DM1** packet type. The interpretation of the value for the Packet_Type command parameter will depend on the Link_Type command parameter returned in the Connection Complete event at the connection setup. Multiple packet types may be specified for the Packet_Type command parameter by bitwise OR operation of the different packet types. See Table 151 and Table 152. For a definition of the different packet types, see Clause 8.

The host should enable as many packet types as possible for the LM to perform efficiently. However, the host must not enable packet types that the local device does not support.

To change an eSCO connection, use the Setup Synchronous Connection command.

Table 151—Change Connection Packet Type command

Command	OCF	Command parameters	Return parameters
HCI_Change_Connection_ Packet_Type	0x000F	Connection_Handle, Packet_Type	None

Table 152—Change Connection Packet Type command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used for transmitting and receiving voice or data. Returned from creating a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Packet_Type for ACL links	2 octets	0x0001	Reserved for future use.
		0x0002	Reserved for future use.
		0x0004	Reserved for future use.
		0x0008[a]	**DM1**
		0x0010	**DH1**
		0x0020	Reserved for future use.
		0x0040	Reserved for future use.
		0x0080	Reserved for future use.
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	**DM3**

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 152—Change Connection Packet Type command parameters *(continued)*

Parameter	Size	Value	Parameter description
Packet_Type for ACL links *(continued)*		0x0800	**DH3**
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	**DM5**
		0x8000	**DH5**
Packet_Type for SCO links	2 octets	0x0001	Reserved for future use.
		0x0002	Reserved for future use.
		0x0004	Reserved for future use.
		0x0008	Reserved for future use.
		0x0010	Reserved for future use.
		0x0020	**HV1**
		0x0040	**HV2**
		0x0080	**HV3**
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	Reserved for future use.
		0x0800	Reserved for future use.
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	Reserved for future use.
		0x8000	Reserved for future use.

[a]This bit will be interpreted as set to 1 by link controllers based on IEEE Std 802.15.1-2005 or newer.

When the controller receives the Change Connection Packet Type command, the controller sends the Command Status event to the host. In addition, when the LM determines the packet type has been changed for the connection, the controller on the local device will send a Connection Packet Type Changed event to the host. This will be done at the local side only.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Connection Packet Type Changed event will indicate that this command has been completed.

11.7.1.15 Authentication Requested command

The Authentication Requested command is used to try to authenticate the remote device associated with the specified connection handle. On an authentication failure, the controller or LM shall not automatically detach the link. The host is responsible for issuing a Disconnect command to terminate the link if the action is appropriate. See Table 153 and Table 154.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 153—Authentication Requested command

Command	OCF	Command parameters	Return parameters
HCI_Authentication_ Requested	0x0011	Connection_Handle	None

Table 154—Authentication requested command parameter

Parameter	Size	Value	Parameter description
Connection _Handle	2 octets	0xXXXX	Connection handle to be used to set up authentication for all connection handles with the same device endpoint as the specified connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

The Connection_Handle command parameter is used to identify the other device that forms the connection. The connection handle should be a connection handle for an ACL connection.

When the controller receives the Authentication Requested command, it sends the Command Status event to the host. The Authentication Complete event will occur when the authentication has been completed for the connection and is indication that this command has been completed.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Authentication Complete event will indicate that this command has been completed.

When the local or remote controller does not have a link key for the specified connection handle, it will request the link key from its host before the local host finally receives the Authentication Complete event.

11.7.1.16 Set Connection Encryption command

The Set Connection Encryption command is used to enable and disable the link-level encryption.

The Connection_Handle command parameter is used to identify the other device that forms the connection. The connection handle should be a connection handle for an ACL connection.

While the encryption is being changed, all ACL traffic must be turned off for all connection handles associated with the remote device. See Table 155 and Table 156.

Table 155—Set Connection Encryption command

Command	OCF	Command parameters	Return parameters
HCI_Set_Connection_ Encryption	0x0013	Connection_Handle, Encryption_Enable	None

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 156—Set Connection Encryption command parameters

Parameters	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to enable/disable the link-layer encryption for all connection handles with the same device endpoint as the specified connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Encryption_ Enable	1 octet	0x00	Turn link-level encryption off.
		0x01	Turn link-level encryption on.

When the controller receives the Set Connection Encryption command, the controller sends the Command Status event to the host. When the LM has completed enabling/disabling encryption for the connection, the controller on the local device will send an Encryption Change event to the host, and the controller on the remote device will also generate an Encryption Change event.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Encryption Change event will indicate that this command has been completed.

11.7.1.17 Change Connection Link Key command

The Change Connection Link Key command is used to force both devices of a connection associated with the connection handle to generate a new link key. The link key is used for authentication and encryption of connections.

The Connection_Handle command parameter is used to identify the other device forming the connection. The connection handle should be a connection handle for an ACL connection. See Table 157 and Table 158.

Table 157—Change Connection Link Key command

Command	OCF	Command parameters	Return parameters
HCI_Change_Connection_ Link_Key	0x0015	Connection_Handle	None

Table 158—Change Connection Link Key command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the controller receives the Change Connection Link Key command, the controller sends the Command Status event to the host. When the LM has changed the link key for the connection, the controller on the local device will send a Link Key Notification event and a Change Connection Link Key Complete event to the host, and the controller on the remote device will also generate a Link Key Notification event. The Link Key Notification event indicates that a new connection link key is valid for the connection. No

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Change Connection Link Key Complete event will indicate that this command has been completed.

11.7.1.18 Master Link Key command

The Master Link Key command is used to force the device that is master of the piconet to use the temporary link key of the master device or the semi-permanent link keys. The temporary link key is used for encryption of broadcast messages within a piconet, and the semi-permanent link keys are used for private encrypted point-to-point communication. The Key_Flag command parameter is used to indicate which link key (temporary link key of the master or the semi-permanent link keys) shall be used. See Table 159 and Table 160.

Table 159—Master Link Key command

Command	OCF	Command parameters	Return parameters
HCI_Master_Link_Key	0x0017	Key_Flag	None

Table 160—Master Link Key command parameter

Parameter	Size	Value	Parameter description
Key_Flag	1 octet	0x00	Use semi-permanent link keys.
		0x01	Use temporary link key.

When the controller receives the Master Link Key command, the controller sends the Command Status event to the host. When the LM has changed link key, the controller on both the local and the remote device will send a Master Link Key Complete event to the host. The connection handle on the master side will be a connection handle for one of the existing connections to a slave. On the slave side, the connection handle will be a connection handle to the initiating master.

The Master Link Key Complete event contains the status of this command.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Master Link Key Complete event will indicate that this command has been completed.

11.7.1.19 Remote Name Request command

The Remote Name Request command is used to obtain the user-friendly name of another device. The user-friendly name is used to enable the user to distinguish one device from another. The BD_ADDR command parameter is used to identify the device for which the user-friendly name is to be obtained. The Page_Scan_Repetition_Mode parameter specifies the page scan repetition mode supported by the remote device with the BD_ADDR. This is the information that was acquired during the inquiry process. The Clock_Offset parameter is the difference between its own clock and the clock of the remote device with BD_ADDR. Only bits 2 through 16 of the difference are used, and they are mapped to this parameter as bits 0 through 14, respectively. A Clock Offset Valid flag, located in bit 15 of the Clock_Offset command parameter, is used to indicate if the clock offset is valid or not. See Table 161 and Table 162.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 161—Remote Name Request command

Command	OCF	Command parameters	Return parameters
HCI_Remote_Name_Request	0x0019	BD_ADDR, Page_Scan_Repetition_Mode, Reserved, Clock_Offset	None

Table 162—Remote Name Request command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR for the device whose name is requested.
Page_Scan_ Repetition_ Mode	1 octet	0x00	R0
		0x01	R1
		0x02	R2
		0x03–0xFF	Reserved.
Reserved	1 octet	0x00	Reserved, must be set to 0x00. See 11.8.1.
Clock_Offset	1 octet	Bit 14.0	Bit 16–2 of CLK_{slave}–CLK_{master}.
		Bit 15	Clock Offset Valid flag Invalid clock offfset = 0 Valid clock offset = 1

If no connection exists between the local device and the device corresponding to the BD_ADDR, a temporary link-layer connection will be established to obtain the name of the remote device.

When the controller receives the Remote Name Request command, the controller sends the Command Status event to the host. When the LM has completed the LMP messages to obtain the remote name, the controller on the local device will send a Remote Name Request Complete event to the host.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Remote Name Request Complete event will indicate that this command has been completed.

11.7.1.20 Remote Name Request Cancel command

This command is used to request cancellation of the ongoing remote name request process, which was started by the Remote Name Request command. See Table 163, Table 164, and Table 165.

Table 163—Remote Name Request Cancel command

Command	OCF	Command parameters	Return parameters
HCI_Remote_Name_Request_Cancel	0x001A	BD_ADDR	Status, BD_ADDR

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 164—Remote Name Request Cancel command parameter

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR of the device for which the Remote Name Request command was issued before and that is the subject of this cancellation request.

Table 165—Remote Name Request Cancel command parameters returned

		Value	Parameter description
Status	1 octet	0x00	Remote Name Request Cancel command succeeded
		0x01–0xff	Remote Name Request Cancel command failed. See 10.3 for list of error codes.
BD_ADDR	6	0xXXXXXXXXXXXX	BD_ADDR of the device for which the Remote Name Request command was issued before and that was the subject of this cancellation request.

When the Remote Name Request Cancel command has completed, a Command Complete event shall be generated.

If the Remote Name Request Cancel command is sent to the controller without a preceding Remote Name Request command to the same device, the controller will return a Command Complete event with the error code *invalid HCI command parameters* (0x12).

The error codes *unspecified error* (0x1F) and *command disallowed* (0x0C) may be used if the controller does not support this command.

The Remote Name Request Complete event for the corresponding Remote Name Request Command shall always be sent. The Remote Name Request Complete event shall be sent after the Command Complete event for the Remote Name Request Cancel command. If the cancellation was successful, the Remote Name Request Complete event will be generated with the error code *unknown connection identifier* (0x02).

11.7.1.21 Read Remote Supported Features command

This command requests a list of the supported features for the remote device identified by the Connection_Handle parameter. The connection handle must be a connection handle for an ACL connection. The Read Remote Supported Features Complete event will return a list of the LMP features. See Table 166 and Table 167. For details see Clause 9.

Table 166—Read Remote Supported Features command

Command	OCF	Command parameters	Return parameters
HCI_Read_Remote_Supported_Features	0x001B	Connection_Handle	None

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 167—Read Remote Supported Features command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's LMP-supported features list to get. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the controller receives the Read Remote Supported Features command, the controller sends the Command Status event to the host. When the LM has completed the LMP messages to determine the remote features, the controller on the local device will send a Read Remote Supported Features Complete event to the host. The Read Remote Supported Features Complete event contains the status of this command and parameters describing the supported features of the remote device.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Read Remote Supported Features Complete event will indicate that this command has been completed.

11.7.1.22 Read Remote Extended Features command

The Read Remote Extended Features command returns the requested page of the extended LMP features for the remote device identified by the specified connection handle. The connection handle must be the connection handle for an ACL connection. This command is available only if the extended features feature is implemented by the remote device. The Read Remote Extended Features Complete event will return the requested information. See Table 168 and Table 169. For details see Clause 9.

Table 168—Read Remote Extended Features command

Command	OCF	Command parameters	Return parameters
HCI_Read_Remote_Extended_Features	0x001C	Connection_Handle, Page_Number	None

Table 169—Read Remote Extended Features command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	The connection handle identifying the remote device for which extended features information is required. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Page_ Number	1 octet	0x00	Requests the normal LMP features as returned by Read Remote Supported Features command.
		0x01–0xFF	Returns the corresponding page of features.

When the controller receives the Read Remote Extended Features command, the controller sends the Command Status command to the host. When the LM has completed the LMP sequence to determine the remote extended features, the controller on the local device will generate a Read Remote Extended Features

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Complete event to the host. The Read Remote Extended Features Complete event contains the page number and the remote features returned by the remote device.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Read Remote Extended Features Complete event will indicate that this command has been completed.

11.7.1.23 Read Remote Version Information command

This command will obtain the values for the version information for the remote device identified by the Connection_Handle parameter. The connection handle must be a connection handle for an ACL connection. See Table 170 and Table 171.

Table 170—Read Remote Version Information command

Command	OCF	Command parameters	Return parameters
HCI_Read_Remote_Version_ Information	0x001D	Connection_Handle	None

Table 171—Read Remote Version Information command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's version information to get. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the controller receives the Read Remote Version Information command, the controller sends the Command Status event to the host. When the LM has completed the LMP messages to determine the remote version information, the controller on the local device will send a Read Remote Version Information Complete event to the host. The Read Remote Version Information Complete event contains the status of this command and parameters describing the version and subversion of the LMP used by the remote device.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Read Remote Version Information Complete event will indicate that this command has been completed.

11.7.1.24 Read Clock Offset command

Both the system clock and the clock offset to a remote device are used to determine what hopping frequency is used by a remote device for page scan. This command allows the host to read clock offset to remote devices. The clock offset can be used to speed up the paging procedure when the local device tries to establish a connection to a remote device, e.g., when the local host has issued Create Connection or Remote Name Request command. The connection handle must be a connection handle for an ACL connection. See Table 172 and Table 173.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 172—Read Clock Offset command

Command	OCF	Command parameters	Return parameters
HCI_Read_Clock_Offset	0x001F	Connection_Handle	None

Table 173—Read Clock Offset command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's Clock_Offset parameter is returned. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the controller receives the Read Clock Offset command, the controller sends the Command Status event to the host. If this command was requested at the master and the LM has completed the LMP messages to obtain the clock offset information, the controller on the local device will send a Read Clock Offset Complete event to the host.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Read Clock Offset Complete event will indicate that this command has been completed. If the command is requested at the slave, the LM will immediately send a Command Status event and a Read Clock Offset Complete event to the host, without an exchange of LMP PDU.

11.7.1.25 Read LMP Handle command

This command will read the current LMP handle associated with the connection handle. The connection handle must be a SCO or eSCO handle. If the connection handle is a SCO connection handle, then this command shall read the LMP SCO handle for this connection. If the connection handle is an eSCO connection handle, then this command shall read the LMP eSCO handle for this connection. See Table 174, Table 175, and Table 176.

Table 174—Read LMP Handle command

Command	OCF	Command parameters	Return parameters
HCI_Read_LMP_Handle	0x0020	Connection_Handle	Status, Connection_Handle, LMP_Handle, Reserved

Table 175—Read LMP Handle command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify which connection to be used for reading the LMP handle. This must be a synchronous handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 176—Read LMP Handle command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read LMP Handle command succeeded.
		0x01–0xFF	Read LMP Handle command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the LMP handle has been read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
LMP_Handle	1 octet	0xXX	The LMP handle that is associated with this connection handle. For a synchronous handle, this would be the LMP synchronous handle used when negotiating the synchronous connection in the LM.
Reserved	4 octets	0xXXXXXXXX	This parameter is reserved, must to set to zero.

When the Read LMP Handle command has completed, a Command Complete event will be generated.

11.7.1.26 Setup Synchronous Connection command

The Setup Synchronous Connection command adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection_Handle parameter specified. If the connection handle refers to an ACL link, a new synchronous logical transport will be added. If the connection handle refers to an already existing synchronous logical transport (eSCO only), this link will be modified. The parameters are specified per connection. This synchronous connection can be used to transfer synchronous voice at 64 kb/s or transparent synchronous data.

When used to set up a new synchronous logical transport, the Connection_Handle parameter shall specify an ACL connection with which the new synchronous connection will be associated. The other parameters relate to the negotiation of the link and may be reconfigured during the lifetime of the link. The TX and RX bandwidth specify how much bandwidth shall be available for transmitting and for receiving data. While in many cases the RX and TX bandwidth parameters may be equal, they may be different. The latency specifies an upper limit to the time in milliseconds between the eSCO (or SCO) instants, plus the size of the retransmission window, plus the length of the reserved synchronous slots for this logical transport. The content format specifies the settings for voice or transparent data on this connection. The retransmission effort specifies the extra resources that are allocated to this connection if a packet may need to be retransmitted. The Retransmission_Effort parameter shall be set to indicate the required behavior, or to “do not care.”

When used to modify an existing synchronous logical transport, the Transmit_Bandwidth, Receive_ Bandwidth, and Voice_Setting parameters shall be set to the same values as were used during the initial setup. The Packet_Type, Retransmission_Effort, and Max_Latency parameters may be modified.

The Packet_Type parameter is a bit map specifying which packet types the LM shall accept in the negotiation of the link parameters. Multiple packet types are specified by bitwise OR of the packet type codes in the table. At least one packet type must be specified for each negotiation. It is recommended to enable as many packet types as possible. Note that it is allowed to enable packet types that are not supported by the local device.

A connection handle for the new synchronous connection will be returned in the Synchronous Connection Complete event.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The LM may choose any combination of packet types, timing, and retransmission window sizes that satisfy the parameters given. This may be achieved by using more frequent transmissions of smaller packets. The LM may choose to set up either a SCO or an eSCO connection, if the parameters allow, using the corresponding LMP sequences.

To modify a SCO connection, use the Change Connection Packet Type command.

If the lower layers cannot achieve the exact TX and RX bandwidth requested subject to the other parameters, then the link shall be rejected.

A synchronous connection may be created only when an ACL connection already exists and when it is not in PARK state. See Table 177 and Table 178.

Table 177—Setup Synchronous Connection command

Command	OCF	Command parameters	Return parameters
HCI_Setup_Synchronous_Connection	0x0028	Connection_Handle Transmit_Bandwidth Receive_Bandwidth Max_Latency Voice_Setting Retransmission_Effort Packet_Type	None

Table 178—Setup Synchronous Connection command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle for the ACL connection being used to create a synchronous connection or for the existing connection that shall be modified. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Transmit_ Bandwidth	4 octets	0xXXXXXXXX	TX bandwidth in octets per second.
Receive_ Bandwidth	4 octets	0xXXXXXXXX	RX bandwidth in octets per second.
Max_Latency	2 octets	0x0000–0x0003	Reserved
		0x0004-0xFFFE	A value, in milliseconds, representing the upper limit of the sum of the synchronous interval and the size of the eSCO window.
		0xFFFF	Do not care.
Voice_ setting	2 octets	See 11.6.12	

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 178—Setup Synchronous Connection command parameters *(continued)*

Parameter	Size	Value	Parameter description
Retransmission_ Effort	1 octet	0x00	No retransmissions
		0x01	At least one retransmission, optimize for power consumption.
		0x02	At least one retransmission, optimize for link quality.
		0xFF	Do not care.
		0x03–0xFE	Reserved.
Packet_ Type	2 octets	0x0001	**HV1**
		0x0002	**HV2**
		0x0004	**HV3**
		0x0008	**EV3**
		0x0010	**EV4**
		0x0020	**EV5**
		0x0040	Reserved for future use.
		0x0080	Reserved for future use.
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	Reserved for future use.
		0x0800	Reserved for future use.
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	Reserved for future use.
		0x8000	Reserved for future use.

When the controller receives the Setup Synchronous Connection command, it sends the Command Status event to the host. In addition, when the LM determines the connection is established, the local controller will send a Synchronous Connection Complete event to the local host, and the remote controller will send a Synchronous Connection Complete event or a Connection Complete event to the remote host. The Synchronous Connection Complete event contains the connection handle if this command is successful.

If this command is used to change the parameters of an existing eSCO link, the Synchronous Connection Changed event is sent to both hosts. In this case, no Connection Complete event or Connection Request event will be sent to either host. These commands cannot be used to change the parameters of an SCO link.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Synchronous Connection Complete event will indicate that this command has been completed.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.1.27 Accept Synchronous Connection Request command

The Accept Synchronous Connection Request command is used to accept an incoming request for a synchronous connection and to inform the local LM about the acceptable parameter values for the synchronous connection. The command shall be issued only after a Connection Request event with link type SCO or eSCO has occurred. The Connection Request event contains the BD_ADDR of the device requesting the connection. The decision to accept a connection must be taken before the connection accept timeout expires on the local device.

The parameter set of the Accept Synchronous Connection Request command is the same as for the Setup Synchronous Connection command. The Transmit_Bandwidth and Receive_Bandwidth parameter values are required values for the new link and shall be met. The Max_Latency parameter is an upper bound to the acceptable latency for the link, as defined in 11.7.1.26, and shall not be exceeded. The Content_Format parameter specifies the encoding in the same way as in the Setup_Synchronous_Connection command and shall be met. The Retransmission_Effort parameter shall be set to indicate the required behavior or to "do not care." The Packet_Type parameter is a bit mask specifying the synchronous packet types that are allowed on the link and shall be met. The reserved bits in the Packet_Type parameter shall be set to one.

If the link type of the incoming request is SCO, then only the Transmit_Bandwidth, Max_Latency, Content_Format, and Packet_Type parameters are valid.

If the Connection Request event is masked away and the controller is not set to automatically accept this connection attempt, the controller will automatically reject it. If the controller is set to automatically accept the connection attempt, the LM should assume default parameters. In that case, the Synchronous Connection Complete event shall be generated, unless masked away. See Table 179 and Table 180.

Table 179—Accept Synchronous Connection Request command

Command	OCF	Command parameters	Return parameters
HCI_Accept_ Synchronous_ Connection_Request	0x0029	BD_ADDR Transmit_Bandwidth Receive_Bandwidth Max_Latency Content_Format Retransmission_Effort Packet_Type	None

Table 180—Accept Synchronous Connection Request command parameter

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXX XXXXX	BD_ADDR of the device requesting the connection.
Transmit_Bandwidth	4 octets	0x00000000–0xFFFFFFFE	Maximum possible TX bandwidth in octets per second.
		0xFFFFFFFF	Do not care. (default)
Receive_Bandwidth	4 octets	0x00000000–0xFFFFFFFE	Maximum possible RX bandwidth in octets per second.
		0xFFFFFFFF	Do not care. (default)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 180—Accept Synchronous Connection Request command parameter *(continued)*

Parameter	Size	Value	Parameter description
Max_Latency	2 octets	0x0000–0x0003	Reserved.
		0x0004–0xFFFE	A value, in milliseconds, representing the upper limit of the sum of the synchronous interval and the size of the eSCO window.
		0xFFFF	Do not care. (default)
Content_Format	2 octets	Default	When links are automatically accepted, the values written by the Write_Voice_Setting parameter are used.
		00XXXXXXXX	Input coding: linear.
		01XXXXXXXX	Input coding: μ-law.
		10XXXXXXXX	Input coding: A-law.
		11XXXXXXXX	Reserved for future use.
		XX00XXXXXX	Input data format: ones complement.
		XX01XXXXXX	Input data format: twos complement.
		XX10XXXXXX	Input data format: sign-magnitude.
		XX11XXXXXX	Input data format: unsigned.
		XXXX0XXXXX	Input sample size: 8 bit (only for linear PCM).
		XXXX1XXXXX	Input sample size: 16 bit (only for linear PCM).
		XXXXXnnnXX	Linear PCM bit position: number of bit positions that MSB of sample is away from starting at MSB (only for linear PCM).
		XXXXXXXX00	Air coding format: CVSD.
		XXXXXXXX01	Air coding format: μ-law.
		XXXXXXXX10	Air coding format: A-law.
		XXXXXXXX11	Air coding format: transparent data.
Retransmission_ Effort	1 octet	0x00	No retransmissions.
		0x01	At least one retransmission, optimize for power consumption.
		0x02	At least one retransmission, optimize for link quality.
		0x03–0xFE	Reserved.
		0xFF	Do not care.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 180—Accept Synchronous Connection Request command parameter *(continued)*

Parameter	Size	Value	Parameter description
Packet_Type	2 octets	0x0001	**HV1**
		0x0002	**HV2**
		0x0004	**HV3**
		0x0008	**EV3**
		0x0010	**EV4**
		0x0020	**EV5**
		0x0040	Reserved for future use.
		0x0080	Reserved for future use.
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	Reserved for future use.
		0x0800	Reserved for future use.
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	Reserved for future use.
		0x8000	Reserved for future use.
		Default:	0xFFFF.

The Accept Synchronous Request command will cause the Command Status event to be sent from the host controller when the host controller starts setting up the connection. When the link setup is complete, the local controller will send a Synchronous Connection Complete event to its host, and the remote controller will send a Connection Complete event or a Synchronous Connection Complete event to the host. The Synchronous Connection Complete will contain the connection handle and the link parameters if the setup is successful. No Command Complete event will be sent by the host controller as the result of this command.

11.7.1.28 Reject Synchronous Connection Request command

The Reject Synchronous Connection Request command is used to decline an incoming request for a synchronous link. It shall be issued only after a Connection Request event with link type equal to SCO or eSCO has occurred. The Connection Request event contains the BD_ADDR of the device requesting the connection. The Reason parameter will be returned to the initiating host in the Status parameter of the Synchronous Connection Complete event on the remote side. See Table 181 and Table 182.

Table 181—Reject Synchronous Connection Request command

Command	OCF	Command parameters	Return parameters
HCI_Reject_Synchronous_ Connection_Request	0x002A	BD_ADDR Reason	None

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 182—Reject Synchronous Connection Request command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXX XXXXX	BD_ADDR of the device requesting the connection.
Reason	1 octet	0x0D–0x0F	See 10.3 for error codes and description.

When the host controller receives the Reject Synchronous Connection Request command, it sends a Command Status event to the host. When the setup is terminated, the local controller will send a Synchronous Connection Complete event to its host, and the remote controller will send a Connection Complete event or a Synchronous Connection Complete event to the host with the reason code from this command. No Command Complete event will be sent by the host controller to indicate that this command has been completed.

11.7.2 Link policy commands

The link policy commands provide methods for the host to affect how the LM manages the piconet. When link policy commands are used, the LM still controls how piconets and scatternets are established and maintained, depending on adjustable policy parameters. These policy commands modify the LM behavior that can result in changes to the link-layer connections with IEEE 802.15.1-2005 remote devices.

Only one ACL connection can exist between two devices; therefore, there can be only one ACL HCI connection handle for each physical link-layer connection. The controller provides policy adjustment mechanisms to provide support for a number of different policies. This capability allows one IEEE 802.15.1-2005 module to be used to support many different usage models, and the same module can be incorporated in many different types of devices. For the link policy commands, the OGF is defined as 0x02.

11.7.2.1 HOLD Mode command

The HOLD Mode command is used to alter the behavior of the LM and have it place the ACL BB connection associated by the specified connection handle into HOLD mode. The Hold_Mode_Max_Interval and Hold_Mode_Min_Interval command parameters specify the length of time the host wants to put the connection into HOLD mode. The local and remote devices will negotiate the length in HOLD mode. The Hold_Mode_Max_Interval parameter is used to specify the maximum length of the hold interval for which the host may actually enter into HOLD mode after negotiation with the remote device. The hold interval defines the amount of time between when the HOLD mode begins and when the HOLD mode is completed. The Hold_Mode_Min_Interval parameter is used to specify the minimum length of the hold interval for which the host may actually enter into HOLD mode after the negotiation with the remote device. Therefore, the HOLD mode minimum interval cannot be greater than the HOLD mode maximum interval. The controller will return the actual hold interval in the Interval parameter of the Mode Change event if the command is successful. This command enables the host to support a low-power policy for itself or several other devices and allows the devices to enter inquiry scan, page scan, and a number of other possible actions.

The connection handle cannot be of the SCO or eSCO link type. If the host sends data to the controller with a connection handle corresponding to a connection in HOLD mode, the controller will keep the data in its buffers until either the data can be transmitted (the HOLD mode has ended) or a flush, a flush timeout, or a disconnection occurs. This is valid even if the host has not yet been notified of the HOLD mode through a Mode Change event when it sends the data.

It is not valid for an HCI data packet sent from the host to the controller on the master side where the connection handle is a connection handle used for broadcast and the Broadcast flag is set to “active broadcast” or “piconet broadcast.” The broadcast data will then never be received by slaves in HOLD mode.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The Hold_Mode_Max_Interval parameter value shall be less than the Link Supervision Timeout configuration parameter value. See Table 183 and Table 184.

Table 183—HOLD Mode command

Command	OCF	Command parameters	Return parameters
HCI_HOLD_Mode	0x0001	Connection_Handle, Hold_Mode_Max_Interval, Hold_Mode_Min_Interval	None

Table 184—HOLD Mode command parameters

Parameter	Size	Value	Parameter description
Connection_Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Hold_Mode_Max_ Interval	2 octets	*N* = 0xXXXX	Maximum acceptable number of BB slots to wait in HOLD mode. Time length of the hold = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0002–0xFFFE; only even values are valid Time range: 1.25 ms to 40.9 s Mandatory range: 0x0014–0x8000
Hold_Mode_Min_ Interval	2 octets	*N* = 0xXXXX	Minimum acceptable number of BB slots to wait in HOLD mode. Time length of the hold = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0002–0xFF00; only even values are valid Time range: 1.25 ms to 40.9 s Mandatory range: 0x0014–0x8000

The controller sends the Command Status event for this command to the host when it has received the HOLD Mode command. The Mode Change event will occur when the HOLD mode has started, and the Mode Change event will occur again when the HOLD mode has completed for the specified connection handle. The Mode Change event signaling the end of the HOLD mode is an estimation of the HOLD mode ending if the event is for a remote device.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed. If an error occurs after the Command Status event has occurred, then the status in the Mode Change event will indicate the error.

11.7.2.2 SNIFF Mode command

The SNIFF Mode command is used to alter the behavior of the LM and have it place the ACL BB connection associated with the specified connection handle into SNIFF mode. The Connection_Handle command parameter is used to identify which ACL link connection is to be placed in SNIFF mode. The Sniff_Max_Interval and Sniff_Min_Interval command parameters are used to specify the requested acceptable maximum and minimum periods in SNIFF mode. The sniff minimum interval shall not be greater than the sniff maximum interval. The sniff interval defines the amount of time between each consecutive sniff period. The controller will return the actual sniff interval in the Interval parameter of the Mode Change event if the command is successful. For a description of the meaning of the Sniff_Attempt and Sniff_Timeout

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

parameters, see 8.8.7. Sniff attempt is there called $N_{sniff\_attempt}$ and sniff timeout is called $N_{sniff\_timeout}$. This command enables the host to support a low-power policy for itself or several other devices and allows the devices to enter inquiry scan, page scan, and a number of other possible actions.

In addition, the connection handle cannot be one of the synchronous link types. If the host sends data to the controller with a connection handle corresponding to a connection in SNIFF mode, the controller will keep the data in its buffers until either the data can be transmitted or a flush, a flush timeout, or a disconnection occurs. This is valid even if the host has not yet been notified of the SNIFF mode through a Mode Change event when it sends the data.

It is possible for the master to transmit data to a slave without exiting SNIFF mode (see description in 8.8.7).

It is not valid for an HCI data packet sent from the host to the controller on the master side where the connection handle is a connection handle used for broadcast and the Broadcast flag is set to "active broadcast" or "piconet broadcast." In that case, the broadcast data will be received only by a slave in SNIFF mode if that slave happens to listen to the master when the broadcast is made.

The Sniff_Max_Interval parameter value shall be less than the Link Supervision Timeout configuration parameter value. See Table 185 and Table 186.

Table 185—SNIFF Mode command

Command	OCF	Command parameters	Return parameters
HCI_SNIFF_Mode	0x0003	Connection_Handle, Sniff_Max_Interval, Sniff_Min_Interval, Sniff_Attempt, Sniff_Timeout	None

Table 186—SNIFF mode command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Sniff_Max_ Interval	2 octets	*N* = 0xXXXX	Range: 0x0002–0xFFFE; only even values are valid Mandatory range: 0x0006–0x0540 Time = *N* * 0.625 ms Time range: 1.25 ms to 40.9 s
Sniff_Min_ Interval	2 octets	*N* = 0xXXXX	Range: 0x0002–0xFFFE; only even values are valid Mandatory range: 0x0006–0x0540 Time = *N* * 0.625 ms Time range: 1.25 ms to 40.9 s
Sniff_Attempt	2 octets	*N* = 0xXXXX	Number of BB receive slots for sniff attempt. Length = *N* * 1.25 ms Range for *N*: 0x0001–0x7FFF Time range: 0.625 ms to 40.9 s Mandatory range for controller: 1 to $T_{sniff}/2$

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 186—SNIFF mode command parameters *(continued)*

Parameter	Size	Value	Parameter description
Sniff_Timeout	2 octets	*N* = 0xXXXX	Number of BB receive slots for sniff timeout. Length = *N* * 1.25 ms Range for *N*: 0x0000–0x7FFF Time range: 0–40.9 s Mandatory range for controller: 0–0x0028

The controller sends the Command Status event for this command to the host when it has received the SNIFF Mode command. The Mode Change event will occur when the SNIFF mode has started for the specified connection handle.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed. If an error occurs after the Command Status event has occurred, then the status in the Mode Change event will indicate the error.

11.7.2.3 Exit SNIFF mode command

The Exit SNIFF Mode command is used to end the SNIFF mode for a connection handle that is currently in SNIFF mode. The LM will determine and issue the appropriate LMP commands to remove the SNIFF mode for the associated connection handle. See Table 187 and Table 188.

In addition, the connection handle cannot be one of the synchronous link types.

Table 187—Exit SNIFF Mode command

Command	OCF	Command parameters	Return parameters
HCI_Exit_Sniff_Mode	0x0004	Connection_Handle	None

Table 188—Exit SNIFF Mode command parameters

Parameter	Size	Value	Parameter description
Connection_Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

A Command Status event for this command will occur when the controller has received the Exit SNIFF Mode command. The Mode Change event will occur when the SNIFF mode has ended for the specified connection handle.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.2.4 PARK State command

The PARK State command is used to alter the behavior of the LM and to have the LM place the BB connection associated by the specified connection handle into PARK state. The Connection_Handle command parameter is used to identify which connection is to be placed in PARK state. The connection handle must be a connection handle for an ACL connection. The beacon interval command parameters specify the acceptable length of the interval between beacons. However, the remote device may request shorter interval. The Beacon_Max_Interval parameter specifies the acceptable longest length of the interval between beacons. The Beacon_Min_Interval parameter specifies the acceptable shortest length of the interval between beacons. Therefore, the beacon minimum interval cannot be greater than the beacon maximum interval. The controller will return the actual beacon interval in the Interval parameter of the Mode Change event if the command is successful. This command enables the host to support a low-power policy for itself or several other devices, allows the devices to enter inquiry scan and page scan, provides support for large number of devices in a single piconet, and allows the devices to enter a number of other possible activities.

When the host issues the PARK State command, no connection handles for synchronous connections are allowed to exist to the remote device that is identified by the Connection_Handle parameter. If one or more connection handles for synchronous connections exist to that device, depending on the implementation, a Command Status event or a Mode Change event (following a Command Status event where Status = 0x00) will be returned with the error code *command disallowed* (0x0C).

If the host sends data to the controller with a connection handle corresponding to a parked connection, the controller will keep the data in its buffers until either the data can be transmitted (after unpark) or a flush, a flush timeout, or a disconnection occurs. This is valid even if the host has not yet been notified of PARK state through a Mode Change event when it sends the data.

The above is not valid for an HCI data packet sent from the host to the controller on the master side where the connection handle is a connection handle used for piconet broadcast and the Broadcast flag is set to “piconet broadcast.” In that case, slaves in PARK state will also receive the broadcast data. (If the Broadcast flag is set to “active broadcast,” the broadcast data will usually not be received by slaves in PARK state.)

It is possible for the controller to do an automatic unpark to transmit data and then park the connection again depending on the value of the Link_Policy_Settings parameter (see Write Link Policy Settings command in 11.7.2.10) and depending on whether the implementation supports this (optional feature). The optional feature of automatic unpark/park can also be used for link supervision. Whether Mode Change events are returned at automatic unpark/park if this is implemented is vendor specific. This could be controlled by a vendor-specific HCI command. See Table 189 and Table 190.

Table 189—PARK State command

Command	OCF	Command parameters	Return parameters
HCI_Park_State	0x0005	Connection_Handle, Beacon_Max_Interval, Beacon_Min_Interval	

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 190—PARK State command parameters

Parameter	Size	Value	Parameter description
Connection_Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Beacon_Max_ Interval	2 octets	*N* = 0xXXXX	Range: 0x000E–0xFFFE; only even values are valid Mandatory range: 0x000E–0x1000 Time = *N* * 0.625 ms Time range: 8.75 ms to 40.9 s
Beacon_Min_ Interval	2 octets	*N* = 0xXXXX	Range: 0x000E–0xFFFE; only even values are valid Mandatory range: 0x000E–0x1000 Time = *N* * 0.625 ms Time range: 8.75 ms to 40.9 s

The controller sends the Command Status event for this command to the host when it has received the PARK State command. The Mode Change event will occur when the PARK state has started for the specified connection handle.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed. If an error occurs after the Command Status event has occurred, then the status in the Mode Change event will indicate the error.

11.7.2.5 Exit PARK State command

The Exit PARK State command is used to switch the device from PARK state back to the active mode. This command may be issued only when the device associated with the specified connection handle is in PARK state. The connection handle must be a connection handle for an ACL connection. This function does not complete immediately. See Table 191 and Table 192.

Table 191—Exit PARK state command

Command	OCF	Command parameters	Return parameters
HCI_Exit_Park_State	0x0006	Connection_Handle	

Table 192—Exit PARK state command parameters

Parameter	Size	Value	Parameter description
Connection_Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

A Command Status event for this command will occur when the controller has received the Exit PARK State command. The Mode Change event will occur when PARK state has ended for the specified connection handle.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed.

11.7.2.6 QoS Setup command

The QoS Setup command is used to specify QoS parameters for a connection handle. The connection handle must be a connection handle for an ACL connection. These QoS parameter are the same parameters as L2CAP QoS. For more detail, see Clause 14. This allows the LM to have all of the information about what the host is requesting for each connection. The LM will determine if the QoS parameters can be met. Devices that are both slaves and masters can use this command. When a device is a slave, this command will trigger an LMP request to the master to provide the slave with the specified QoS as determined by the LM. When a device is a master, this command is used to request a slave device to accept the specified QoS as determined by the LM of the master. The Connection_Handle command parameter is used to identify for which connection the QoS request is requested. See Table 193 and Table 194.

Table 193—QoS Setup command

Command	OCF	Command parameters	Return parameters
HCI_QoS_Setup	0x0007	Connection_Handle, Flags, Service_Type, Token_Rate, Peak_Bandwidth, Latency, Delay_Variation	None

Table 194—QoS Setup command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify the connection for the QoS setup. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Flags	1 octet	0x00–0xFF	Reserved for future use.
Service_ Type	1 octet	0x00	No traffic.
		0x01	Best effort.
		0x02	Guaranteed.
		0x03–0xFF	Reserved for future use.
Token_Rate	4 octets	0xXXXXXXXX	Token rate in octets per second.
Peak_ Bandwidth	4 octets	0xXXXXXXXX	Peak bandwidth in octets per second.
Latency	4 octets	0xXXXXXXXX	Latency in microseconds.
Delay_ Variation	4 octets	0xXXXXXXXX	Delay variation in microseconds.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When the controller receives the QoS Setup command, the controller sends the Command Status event to the host. When the LM has completed the LMP messages to establish the requested QoS parameters, the controller on the local device will send a QoS Setup Complete event to the host, and the event may also be generated on the remote side if there was LMP negotiation. The values of the parameters of the QoS Setup Complete event may, however, be different on the initiating and the remote side. The QoS Setup Complete event returned by the controller on the local side contains the status of this command and returned QoS parameters describing the supported QoS for the connection.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the QoS Setup Complete event will indicate that this command has been completed.

11.7.2.7 Role Discovery command

The Role Discovery command is used for a device to determine which role the device is performing for a particular connection handle. The connection handle must be a connection handle for an ACL connection. See Table 195, Table 196, and Table 197.

Table 195—Role Discovery command

Command	OCF	Command parameters	Return parameters
HCI_Role_Discovery	0x0009	Connection_Handle	Status, Connection_Handle, Current_Role

Table 196—Role Discovery command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 197—Role Discovery command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Role Discovery command succeeded,
		0x01–0xFF	Role Discovery command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify on which connection the Role Discovery command was issued. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Current_ Role	1 octet	0x00	Current role is master for this connection handle.
		0x01	Current role is slave for this connection handle.

When the Role Discovery command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.2.8 Switch Role command

The Switch Role command is used for a device to switch the current role the device is performing for a particular connection with another specified device. The BD_ADDR command parameter indicates for which connection the role switch is to be performed. The Role parameter indicates the requested new role that the local device performs. See Table 198 and Table 199.

Table 198—Switch Role command

Command	OCF	Command parameters	Return parameters
HCI_Switch_Role	0x000B	BD_ADDR, Role	None

Table 199—Switch Role command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR for the connected device with which a role switch is to be performed.
Role	1 octet	0x00	Change own role to master for this BD_ADDR.
		0x01	Change own role to slave for this BD_ADDR.

The BD_ADDR command parameter must specify a device for which a connection already exists.

If there is an SCO connection between the local device and the device identified by the BD_ADDR parameter, an attempt to perform a role switch shall be rejected by the local device.

If the connection between the local device and the device identified by the BD_ADDR parameter is placed in SNIFF mode, an attempt to perform a role switch will be rejected by the local device.

A Command Status event for this command will occur when the controller has received the Switch Role command. When the role switch is performed, a Role Change event will occur to indicate that the roles have been changed and will be communicated to both hosts.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, only the Role Change event will indicate that this command has been completed.

11.7.2.9 Read Link Policy Settings command

This command will read the link policy setting for the specified connection handle. The connection handle must be a connection handle for an ACL connection. See Table 200, Table 201, and Table 202. For details, see 11.6.19.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 200—Read Link Policy Settings command

Command	OCF	Command parameters	Return parameters
HCI_Read_Link_Policy_ Settings	0x000C	Connection_Handle	Status, Connection_Handle Link_Policy_Settings

Table 201—Read Link Policy Settings command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 202—Read Link Policy Settings command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Link Policy Settings command succeeded.
		0x01–0xFF	Read Link Policy Settings command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Link_ Policy_ Settings	2 octets	0x0000	Disable all LM modes. (default)
		0x0001	Enable role switch.
		0x0002	Enable HOLD mode.
		0x0004	Enable SNIFF mode.
		0x0008	Enable PARK state.
		0x0010–0x8000	Reserved for future use.

When the Read Link Policy Settings command has completed, a Command Complete event will be generated.

11.7.2.10 Write Link Policy Settings command

This command will write the link policy setting for the specified connection handle. The connection handle must be a connection handle for an ACL connection. See 11.6.19. The default value is the value set by the Write Default Link Policy Settings command. See Table 203, Table 204, and Table 205.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 203—Write Link Policy Settings command

Command	OCF	Command parameters	Return parameters
HCI_Write_Link_Policy_ Settings	0x000D	Connection_Handle, Link_Policy_Settings	Status, Connection_Handle

Table 204—Write link policy settings command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Link_ Policy_ Settings	2 octets	0x0000	Disable all LM modes.
		0x0001	Enable role switch.
		0x0002	Enable HOLD mode.
		0x0004	Enable SNIFF mode.
		0x0008	Enable PARK state.
		0x0010–0x8000	Reserved for future use.

Table 205—Write Link Policy Settings command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Link Policy Settings command succeeded.
		0x01–0xFF	Write Link Policy Settings command failed. See 10.3 for error codes and description.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the Write Link Policy Settings command has completed, a Command Complete event will be generated.

11.7.2.11 Read Default Link Policy Settings command

This command will read the default link policy setting for all new connections. See Table 206 and Table 207.

Table 206—Read Default Link Policy Settings command

Command	OCF	Command parameters	Return parameters
HCI_Read_Default_Link _Policy_Settings	0x000E	None	Status, Default_Link_Policy_Settings

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 207—Read Default Link Policy Settings command returned parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Link Policy Settings command succeeded.
		0x01–0xFF	Read Link Policy Settings command failed. See 10.3 for list of error codes.
Default_Link_ Policy_Settings	2 octets	0x0000	Disable all LM modes. (default)
		0x0001	Enable role switch.
		0x0002	Enable HOLD mode.
		0x0004	Enable SNIFF mode.
		0x0008	Enable PARK state.
		0x0010–0x8000	Reserved for future use.

Refer to the Link_Policy_Settings configuration parameter for more information. See 11.6.19.

When the Read Default Link Policy Settings command has completed, a Command Complete event will be generated.

11.7.2.12 Write Default Link Policy Settings command

This command will write the default link policy configuration value. The Default_Link_Policy_Settings parameter determines the initial value of the Link_Policy_Settings for all new connections. See Table 208, Table 209, and Table 210.

Table 208—Write Default Link Policy Settings command

Command	OCF	Command parameters	Return parameters
HCI_Write_Default_Link_ Policy_Settings	0x000F	Default_Link_Policy_Settings	Status

Table 209—Write Default Link Policy Settings command parameter

Parameter	Size	Value	Parameter description
Default_ Link_ Policy_ Settings	2 octets	0x0000	Disable all LM modes. (default)
		0x0001	Enable role switch.
		0x0002	Enable HOLD mode.
		0x0004	Enable SNIFF mode.
		0x0008	Enable PARK state.
		0x0010–0x8000	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 210—Write Default Link Policy Settings command returned parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Link Policy Settings command succeeded.
		0x01–0xFF	Write Link Policy Settings command failed. See 10.3 for list of error codes.

Refer to the Link Policy Settings configuration parameter for more information. See 11.6.19.

When the Write Default Link Policy Settings command has completed, a Command Complete event will be generated.

11.7.2.13 Flow Specification command

The Flow Specification command is used to specify the flow parameters for the traffic carried over the ACL connection identified by the connection handle. The connection handle must be a connection handle for an ACL connection. The Connection_Handle command parameter is used to identify for which connection the flow specification is requested. The flow parameters refer to the outgoing or incoming traffic of the ACL link, as indicated by the Flow_Direction field. The Flow Specification command allows the LM to have the parameters of the outgoing as well as the incoming flow for the ACL connection. The flow parameters are defined in 14.5.3. The LM will determine if the flow parameters can be supported. Devices that are both master and slave can use this command. See Table 211 and Table 212.

Table 211—Flow Specification command

Command	OCF	Command parameters	Return parameters
HCI_Flow_Specification	0x0010	Connection_Handle, Flags, Flow_direction, Service_Type, Token_Rate, Token_Bucket_Size, Peak_Bandwidth, Access Latency	

Table 212—Flow specification command parameters

Parameter	Size	Value	Parameter description
Connection_Handle	2 octets	0xXXXX	Connection handle used to identify for which ACL connection the flow is specified. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Flags	1 octet	0x00–0xFF	Reserved for future use.
Flow_direction	1 octet	0x00	Outgoing flow, i.e., traffic send over the ACL connection.
		0x01	Incoming flow, i.e., traffic received over the ACL connection.
		0x02–0xFF	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 212—Flow specification command parameters *(continued)*

Parameter	Size	Value	Parameter description
Service_Type	1 octet	0x00	No traffic.
		0x01	Best effort.
		0x02	Guaranteed.
		0x03–0xFF	Reserved for future use.
Token_Rate	4 octets	0xXXXXXXXX	Token rate in octets per second.
Token_Bucket_Size	4 octets	0xXXXXXXXX	Token bucket size in octets.
Peak_Bandwidth	4 octets	0xXXXXXXXX	Peak bandwidth in octets per second.
Access_Latency	4 octets	0xXXXXXXXX	Latency in microseconds.

When the controller receives the Flow Specification command, the controller sends the Command Status event to the host. When the LM has determined if the flow specification can be supported, the controller on the local device sends a Flow Specification Complete event to the host. The Flow Specification Complete event returned by the controller on the local side contains the status of this command and returned Flow parameters describing the supported QoS for the ACL connection.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Flow Specification Complete event will indicate that this command has been completed.

11.7.3 Controller-BB commands

The controller-BB commands provide access and control to various capabilities of the hardware. These parameters provide control of devices and of the capabilities of the controller, LM, and BB. The host device can use these commands to modify the behavior of the local device. For the HCI controller-BB commands, the OGF is defined as 0x03.

11.7.3.1 Set Event Mask command

The Set Event Mask command is used to control which events are generated by the HCI for the host. If the bit in the Event_Mask parameter is set to a one, then the event associated with that bit will be enabled. The host has to deal with each event that occurs by the devices. The event mask allows the host to control how much it is interrupted. See Table 213, Table 214, and Table 215.

Table 213—Set Event Mask command

Command	OCF	Command parameters	Return parameters
HCI_Set_Event_Mask	0x0001	Event_Mask	Status

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 214—Set Event Mask command parameter

Parameter	Size	Value	Parameter description
Event_Mask	8 octets	0x0000000000000000	No events specified
		0x0000000000000001	Inquiry Complete event
		0x0000000000000002	Inquiry Result event
		0x0000000000000004	Connection Complete event
		0x0000000000000008	Connection Request event
		0x0000000000000010	Disconnection Complete event
		0x0000000000000020	Authentication Complete event
		0x0000000000000040	Remote Name Request Complete event
		0x0000000000000080	Encryption Change event
		0x0000000000000100	Change Connection Link Key Complete event
		0x0000000000000200	Master Link Key Complete event
		0x0000000000000400	Read Remote Supported Features Complete event
		0x0000000000000800	Read Remote Version Information Complete event
		0x0000000000001000	QoS Setup Complete event
		0x0000000000002000	Reserved
		0x0000000000004000	Reserved
		0x0000000000008000	Hardware Error event
		0x0000000000010000	Flush Occurred event
		0x0000000000020000	Role Change event
		0x0000000000040000	Reserved
		0x0000000000080000	Mode Change event
		0x0000000000100000	Return Link Keys event
		0x0000000000200000	PIN Code Request event
		0x0000000000400000	Link Key Request event
		0x0000000000800000	Link Key Notification event
		0x0000000001000000	Loopback Command event
		0x0000000002000000	Data Buffer Overflow event
		0x0000000004000000	Max Slots Change event
		0x0000000008000000	Read Clock Offset Complete event
		0x0000000010000000	Connection Packet Type Changed event
		0x0000000020000000	QoS Violation event
		0x0000000040000000	Page Scan Mode Change event [deprecated]
		0x0000000080000000	Page Scan Repetition Mode Change event

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 214—Set Event Mask command parameter *(continued)*

Parameter	Size	Value	Parameter description
Event_Mask *(continued)*		0x0000000100000000	Flow Specification Complete event
		0x0000000200000000	Inquiry Result With RSSI event
		0x0000000400000000	Read Remote Extended Features Complete event
		0x0000000800000000	Reserved
		0x0000001000000000	Reserved
		0x0000002000000000	Reserved
		0x0000004000000000	Reserved
		0x0000008000000000	Reserved
		0x0000010000000000	Reserved
		0x0000020000000000	Reserved
		0x0000040000000000	Reserved
		0x0000080000000000	Synchronous Connection Complete event
		0x0000100000000000	Synchronous connection changed event
		0xFFFFE00000000000	Reserved for future use
		0x00001FFFFFFFFFFF	**Default** (All events enabled)

Table 215—Set Event Mask command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Set Event Mask command succeeded.
		0x01–0xFF	Set Event Mask command failed. See 10.3 for error codes and description.

When the Set Event Mask command has completed, a Command Complete event will be generated.

11.7.3.2 Reset command

The Reset command will reset the controller and the LM. The reset command shall not affect the used host controller transport layer since the host controller transport layers may have reset mechanisms of their own. After the reset is completed, the current operational state will be lost, the device will enter STANDBY state, and the controller will automatically revert to the default values for the parameters for which default values are defined in the specification. See Table 216 and Table 217.

Table 216—Reset command

Command	OCF	Command parameters	Return parameters
HCI_Reset	0x0003	None	Status

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 217—Reset command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Reset command succeeded, was received, and will be executed.
		0x01–0xFF	Reset command failed. See 10.3 for list of error codes.

The host is not allowed to send additional HCI commands before the Command Complete event related to the Reset command has been received.

When the reset has been performed, a Command Complete event will be generated.

11.7.3.3 Set Event Filter command

The Set Event Filter command is used by the host to specify different event filters. The host may issue this command multiple times to request various conditions for the same type of event filter and for different types of event filters. The event filters are used by the host to specify items of interest, which allow the controller to send only events that interest the host. Only some of the events have event filters. By default (before this command has been issued after power-on or reset), no filters are set, and the Auto Accept flag is off (incoming connections are not automatically accepted). An event filter is added each time this command is sent from the host and the Filter_Condition_Type parameter is not equal to 0x00. (The old event filters will not be overwritten.) To clear all event filters, the Filter_Type parameter value 0x00 is used. The Auto Accept flag will then be set to off. To clear event filters for only a certain filter type, the Filter_Condition_Type parameter value 0x00 is used.

The inquiry result filter allows the controller to filter out Inquiry Result events. The inquiry result filter allows the host to specify that the controller sends inquiry results to the host only if the Inquiry Result event meets one of the specified conditions set by the host. For the inquiry result filter, the host can specify one or more of the following filter condition types:

— Return responses from all devices during the inquiry process
— A device with a specific class of device responded to the inquiry process
— A device with a specific BD_ADDR responded to the inquiry process

The inquiry result filter is used in conjunction with the Inquiry and Periodic Inquiry Mode commands.

The connection setup filter allows the host to specify that the controller sends a Connection Complete or Connection Request event to the host only if the event meets one of the specified conditions set by the host. For the connection setup filter, the host can specify one or more of the following filter condition types:

— Allow connections from all devices
— Allow connections from a device with a specific class of device
— Allow connections from a device with a specific BD_ADDR

For each of these conditions, an Auto Accept flag allows the host to specify what action should be done when the condition is met. The Auto Accept flag allows the host to specify if the incoming connection should be automatically accepted (in which case the controller will send the Connection Complete event to the host when the connection is completed) or if the host should make the decision (in which case the controller will send the Connection Request event to the host to elicit a decision on the connection).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The connection setup filter is used in conjunction with the Read/Write Scan Enable commands. If the local device is in the process of a page scan, the local device is paged by another device that meets one of the conditions set by the host, and the Auto Accept flag is off for this device, then a Connection Request event will be sent to the host by the controller. A Connection Complete event will be sent later on after the host has responded to the incoming connection attempt. In this same example, if the Auto Accept flag is on, then a Connection Complete event will be sent to the host by the controller. (No Connection Request event will be sent in that case.)

The controller will store these filters in volatile memory until the host clears the event filters using the Set Event Filter command or until the Reset command is issued. The number of event filters the controller can store is implementation dependent. If the host tries to set more filters than the controller can store, the controller will return the error code *memory capacity exceeded* and the filter will not be installed.

The Filter_Type parameter set to “clear all filters” has no filter condition types or conditions.

In the condition that a connection is automatically accepted, a Link Key Request event and possibly also a PIN Code Request event and a Link Key Notification event could be sent to the host by the controller before the Connection Complete event is sent.

If there is a contradiction between event filters, the latest set event filter will override older ones. An example is an incoming connection attempt where more than one connection setup filter matches the incoming connection attempt, but the Auto Accept flag has different values in the different filters. See Table 218, Table 219, Table 220, Table 221, and Table 222.

Table 218—Set Event Filter command

Command	OCF	Command parameters	Return parameters
HCI_Set_Event_Filter	0x0005	Filter_Type, Filter_Condition_Type, Condition	Status

Table 219—Set Event Filter command parameters

Parameter	Filter_ Type qualifier value	Filter_ Condition_ Type qualifier value	Size in octets	Value	Parameter description
Filter_Type	—	—	1	0x00	Clear all filters.
				0x01	Inquiry result.
				0x02	Connection setup.
				0x03– 0xFF	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 219—Set Event Filter command parameters *(continued)*

Parameter	Filter_ Type qualifier value	Filter_ Condition_ Type qualifier value	Size in octets	Value	Parameter description
Filter_ Condition_ Type	0x00	—	0		In this case, the Filter_Condition_Type and Condition parameters shall not be given; they shall have a length of 0 octets. Filter_Type should be the only parameter.
	0x01	—	1	0x00	Return responses from all devices during the inquiry process. NOTE—A device may be reported to the host in an Inquiry Result event more than once during an inquiry or inquiry period depending on the implementation (see description in 11.7.1.1 and 11.7.1.3).
				0x01	A device with a specific class of device responded to the inquiry process.
				0x02	A device with a specific BD_ADDR responded to the inquiry process.
				0x03–0xFF	Reserved for future use.
	0x02	—	1	0x00	Allow connections from all devices.
				0x01	Allow connections from a device with a specific class of device.
				0x02	Allow connections from a device with a specific BD_ADDR.
				0x03–0xFF	Reserved for future use.
Condition	0x01	0x01	6	0x000000	Return all devices. (default)
				0xXXXXXX	Class of device of interest.
				0xXXXXXX	Bit mask used to determine which bits of the Class of Device field are “do not care.” Zero-value bits in the mask indicate the “do not care” bits of the Class of Device field.
	0x01	0x02	6	0xXXXXXXXXXX	BD_ADDR of the device of interest

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 219—Set Event Filter command parameters *(continued)*

Parameter	Filter_ Type qualifier value	Filter_ Condition_ Type qualifier value	Size in octets	Value	Parameter description
Condition *(continued)*	0x02	0x00	1	0x01	Do not automatically accept the connection. (Auto Accept flag is off.)
				0x02	Do automatically accept the connection with role switch disabled. (Auto Accept flag is on.)
				0x03	Do automatically accept the connection with role switch enabled. (Auto Accept flag is on.) NOTE—When automatically accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto Accept flag will then get the same effect as if the value had been 0x02.
				0x04–0xFF	Reserved for future use.
	0x02	0x01	7		See Table 220.
	0x02	0x02	7		See Table 221.

Table 220—Condition parameter fields with Filter_Type parameter 0x02 (connection setup) and Filter_Condition_Type parameter 0x01

Parameter field	Size	Value	Description
Class of Device	3 octets	0x000000	Return all devices. (default)
		0xXXXXXX	Class of device of interest.
Class of Device Mask	3 octets	0xXXXXXX	Bit mask used to determine which bits of the Class of Device field are “do not care.” Zero-value bits in the mask indicate the “do not care” bits of the Class of Device field. NOTE—For an incoming SCO connection, if the class of device is unknown, then the connection will be accepted.
Auto Accept (flag)	1 octet	0x01	Do not automatically accept the connection. (Auto Accept flag is off.)
		0x02	Do automatically accept the connection with role switch disabled. (Auto Accept flag is on.)
		0x03	Do automatically accept the connection with role switch enabled. (Auto Accept flag is on.) NOTE—When automatically accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto Accept flag will then get the same effect as if the value had been 0x02.
		0x04–0xFF	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 221—Condition parameter fields with Filter_Type parameter 0x02 (connection setup) and Filter_Condition_Type parameter 0x02

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device of interest.
Auto_Accept _Flag	1 octet	0x01	Do not automatically accept the connection. (Auto Accept flag is off.)
		0x02	Do automatically accept the connection with role switch disabled. (Auto Accept flag is on.)
		0x03	Do automatically accept the connection with role switch enabled. (Auto Accept flag is on.) NOTE—When automatically accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto Accept flag will then get the same effect as if the value had been 0x02.
		0x04–0xFF	Reserved for future use.

Table 222—Set Event Filter command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Set Event Filter command succeeded.
		0x01–0xFF	Set Event Filter command failed. See 10.3 for list of error codes.

A Command Complete event for this command will occur when the controller has enabled the filtering of events. When one of the conditions are met, a specific event will occur.

11.7.3.4 Flush command

The Flush command is used to discard all data that are currently pending for transmission in the controller for the specified connection handle, even if there currently are chunks of data that belong to more than one L2CAP packet in the controller. After this, all data that are sent to the controller for the same connection handle will be discarded by the controller until an HCI data packet with the start Packet Boundary flag (0x02) is received. When this happens, a new transmission attempt can be made. This command will allow higher level software to control how long the BB should try to retransmit a BB packet for a connection handle before all data that are currently pending for transmission in the controller should be flushed. Note that the Flush command is used for ACL connections only. In addition to the Flush command, the automatic flush timers (see 11.7.3.31) can be used to automatically flush the L2CAP packet that is currently being transmitted after the specified flush timer has expired. See Table 223, Table 224, and Table 225.

Table 223—Flush command

Command	OCF	Command parameters	Return parameters
HCI_Flush	0x0008	Connection_Handle	Status, Connection_Handle

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 224—Flush command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify which connection to flush. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 225—Flush command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Flush command succeeded.
		0x01–0xFF	Flush command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify on which connection the flush command was issued. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

The Flush Occurred event will occur once the flush is completed. A Flush Occurred event could be from an automatic flush or could be cause by the host issuing the Flush command. When the Flush command has completed, a Command Complete event will be generated to indicate that the host caused the flush.

11.7.3.5 Read PIN Type command

The Read PIN Type command is used to read the PIN_Type configuration parameter. See Table 226 and Table 227. See also 11.6.13.

Table 226—Read PIN Type command

Command	OCF	Command parameters	Return parameters
HCI_Read_PIN_Type	0x0009	None	Status, PIN_Type

Table 227—Read PIN Type command parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read PIN Type command succeeded.
		0x01–0xFF	Read PIN Type command failed. See 10.3 for list of error codes.
PIN_Type	1 octet	0x00	Variable PIN.
		0x01	Fixed PIN.

When the Read PIN Type command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.6 Write PIN Type command

The Write PIN Type command is used to write the PIN_Type configuration parameter. See Table 228, Table 229, and Table 230. See also 11.6.13.

Table 228—Write PIN Type command

Command	OCF	Command parameters	Return parameters
HCI_Write_PIN_Type	0x000A	PIN_Type	Status

Table 229—Write PIN Type command parameter

Parameter	Size	Value	Parameter description
PIN_Type	1 octet	0x00	Variable PIN.
		0x01	Fixed PIN.

Table 230—Write PIN Type command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write PIN Type command succeeded.
		0x01–0xFF	Write PIN Type command failed. See 10.3 for list of error codes.

When the Write PIN Type command has completed, a Command Complete event will be generated.

11.7.3.7 Create New Unit Key command

The Create New Unit Key command is used to create a new unit key. The hardware will generate a random seed that will be used to generate the new unit key. All new connection will use the new unit key, but the old unit key will still be used for all current connections. See Table 231 and Table 232.

This command will not have any effect for a device that does not use unit keys (i.e., a device that uses only combination keys).

Table 231—Create New Unit Key command

Command	OCF	Command parameters	Return parameters
HCI_Create_New_Unit_Key	0x000B	None	Status

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 232—Create New Unit Key command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Create New Unit Key command succeeded.
		0x01–0xFF	Create New Unit Key command failed. See 10.3 for list of error codes.

When the Create New Unit Key command has completed, a Command Complete event will be generated.

11.7.3.8 Read Stored Link Key command

The Read Stored Link Key command provides the ability to read one or more link keys stored in the controller. The controller can store a limited number of link keys for other devices. Link keys are shared between two devices and are used for all security transactions between the two devices. A host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the controller when needed. The Read_All_Flag parameter is used to indicate if all of the stored link keys should be returned. If Read_All_Flag parameter indicates that all link keys are to be returned, then the BD_ADDR command parameter must be ignored. The BD_ADDR command parameter is used to identify which link key to read. The stored link keys are returned by one or more Return Link Keys events. See Table 233, Table 234, and Table 235. See also 11.6.14.

Table 233—Read Stored Link Key command

Command	OCF	Command parameters	Return parameters
HCI_Read_Stored_ Link_Key	0x000D	BD_ADDR, Read_All_Flag	Status, Max_Num_Keys, Num_Keys_Read

Table 234—Read Stored Link Key command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR for the stored link key to be read.
Read_All_Flag	1 octet	0x00	Return link key for specified BD_ADDR.
		0x01	Return all stored link keys.
		0x02–0xFF	Reserved for future use.

Table 235—Read Stored Link Key command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Stored Link Key command succeeded.
		0x01–0xFF	Read Stored Link Key command failed. See 10.3 for error codes and description.
Max_Num_Keys	2 octets	0xXXXX	Total number of link keys that the controller can store. Range: 0x0000–0xFFFF

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 235—Read Stored Link Key command return parameters *(continued)*

Parameter	Size	Value	Parameter description
Num_Keys_Read	2 octets	0xXXXX	Number of link keys read. Range: 0x0000–0xFFFF

Zero or more instances of the Return Link Keys event will occur after the command is issued. When there are no link keys stored, no Return Link Keys events will be returned. When there are link keys stored, the number of link keys returned in each Return Link Keys event is implementation specific. When the Read Stored Link Key command has completed, a Command Complete event will be generated.

11.7.3.9 Write Stored Link Key command

The Write Stored Link Key command provides the ability to write one or more link keys to be stored in the controller. The controller can store a limited number of link keys for other devices. If no additional space is available in the controller, then no additional link keys will be stored. If space is limited and if all the link keys to be stored will not fit in the limited space, then the order of the list of link keys without any error will determine which link keys are stored. Link keys at the beginning of the list will be stored first. The Num_Keys_Written parameter will return the number of link keys that were successfully stored. If no additional space is available, then the host must delete one or more stored link keys before any additional link keys are stored. The link key replacement algorithm is implemented by the host and not the controller. Link keys are shared between two devices and are used for all security transactions between the two devices. A host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the controller when needed. See Table 236, Table 237, and Table 238. See also 11.6.14

Link keys are stored only by issuing this command.

Table 236—Write Stored Link Key command

Command	OCF	Command parameters	Return parameters
HCI_Write_Stored_Link_Key	0x0011	Num_Keys_To_Write, BD_ADDR[i], Link_Key[i]	Status, Num_Keys_Written

Table 237—Write Stored Link Key command parameters

Parameter	Size	Value	Parameter description
Num_Keys_To_Write	1 octet	0xXX	Number of link keys to write. Range: 0x0–0x0B
BD_ADDR[i]	6 octets * Num_Keys_To_Write	0xXXXXXXXXXXX	BD_ADDR for the associated link key.
Link_Key[i]	16 octets	0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Link key for an associated BD_ADDR.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 238—Write Stored Link Key command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Stored Link Key command succeeded.
		0x01–0xFF	Write Stored Link Key command failed. See 10.3 for list of error codes.
Num_Keys_ Written	1 octets	0xXX	Number of link keys successfully written. Range: 0x00–0x0B

When the Write Stored Link Key command has completed, a Command Complete event will be generated.

11.7.3.10 Delete Stored Link Key command

The Delete Stored Link Key command provides the ability to remove one or more of the link keys stored in the controller. The controller can store a limited number of link keys for other devices. Link keys are shared between two devices and are used for all security transactions between the two devices. The Delete_All_Flag parameter is used to indicate if all of the stored link keys should be deleted. If the Delete_All_Flag parameter indicates that all link keys are to be deleted, then the BD_ADDR command parameter must be ignored. This command provides the ability to negate all security agreements between two devices. The BD_ADDR command parameter is used to identify which link key to delete. If a link key is currently in use for a connection, then the link key will be deleted when all of the connections are disconnected. See Table 239, Table 240, and Table 241. See also 11.6.14.

Table 239—Delete Stored Link Key command

Command	OCF	Command parameters	Return parameters
HCI_Delete_Stored_ Link_Key	0x0012	BD_ADDR, Delete_All_Flag	Status, Num_Keys_Deleted

Table 240—Delete Stored Link Key command parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR for the link key to be deleted.
Delete_All_ Flag	1 octet	0x00	Delete only the link key for specified BD_ADDR.
		0x01	Delete all stored link keys.
		0x02–0xFF	Reserved for future use.

Table 241—Delete Stored Link Key command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Delete Stored Link Key command succeeded.
		0x01–0xFF	Delete Stored Link Key command failed. See 10.3 for error codes and description.
Num_Keys_ Deleted	2 octets	0xXXXX	Number of link keys deleted

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When the Delete Stored Link Key command has completed, a Command Complete event will be generated.

11.7.3.11 Write Local Name command

The Write Local Name command provides the ability to modify the user-friendly name for the device. See Table 242, Table 243, and Table 244. See also 11.6.24.

Table 242—Write Local Name command

Command	OCF	Command parameters	Return parameters
HCI_Write_Local_Name	0x0013	Local_Name	Status

Table 243—Write Local Name command parameter

Parameter	Size	Value	Parameter description
Local_Name	248 octets		A UTF-8 encoded user-friendly descriptive name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.
			Null-terminated zero-length string. (default)

Table 244—Write Local Name command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Local Name command succeeded.
		0x01–0xFF	Write Local Name command failed. See 10.3 for list of error codes.

When the Write Local Name command has completed, a Command Complete event will be generated.

11.7.3.12 Read Local Name command

The Read Local Name command provides the ability to read the stored user-friendly name for the device. See Table 245 and Table 246. See also 11.6.24.

Table 245—Read Local Name command

Command	OCF	Command parameters	Return parameters
HCI_Read_Local_Name	0x0014	None	Status, Local_Name

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 246—Read Local Name command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Local Name command succeeded
		0x01–0xFF	Read Local Name command failed. See 10.3 for list of error codes.
Local_Name	248 octets		A UTF-8 encoded user friendly descriptive name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

When the Read Local Name command has completed, a Command Complete event will be generated.

11.7.3.13 Read Connection Accept Timeout command

This command will read the value for the Connection_Accept_Timeout configuration parameter. See Table 247 and Table 248. See also 11.6.7.

Table 247—Read Connection Accept Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Read_Connection_ Accept_Timeout	0x0015	None	Status, Conn_Accept_Timeout

Table 248—Read Connection Accept Timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Connection Accept Timeout command succeeded.
		0x01–0xFF	Read Connection Accept Timeout command failed. See 10.3 for list of error codes.
Conn_Accept_ Timeout	2 octets	*N* = 0xXXXX	Connection accept timeout measured in number of BB slots. Interval length = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0001–0xB540 Time range: 0.625 ms to 29 s

When the Read Connection Timeout command has completed, a Command Complete event will be generated.

11.7.3.14 Write Connection Accept Timeout command

This command will write the value for the Connection_Accept_Timeout configuration parameter. See Table 249, Table 250, and Table 251. See also 11.6.7.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 249—Write Connection Accept Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Write_Connection_ Accept_Timeout	0x0016	Conn_Accept_Timeout	Status

Table 250—Write Connection Accept Timeout command parameters

Parameter	Size	Value	Parameter description
Conn_Accept_ Timeout	2 octets	*N* = 0xXXXX	Connection accept timeout measured in number of BB slots. Interval length = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0001–0xB540 Time range: 0.625 ms to 29 s Default: N = 0x1FA0 Time = 5.06 s Mandatory range for controller: 0x00A0–0xB540

Table 251—Write Connection Accept Timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Connection Accept Timeout command succeeded.
		0x01–0xFF	Write Connection Accept Timeout command failed. See 10.3 for list of error codes.

When the Write Connection Accept Timeout command has completed, a Command Complete event will be generated.

11.7.3.15 Read Page Timeout command

This command will read the value for the Page_Reply_Timeout configuration parameter. See Table 252 and Table 253. See also 11.6.6.

Table 252—Read Page Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Read_Page_Timeout	0x0017	None	Status, Page_Reply_Timeout

Table 253—Read Page Timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Page Timeout command succeeded.
		0x01–0xFF	Read Page Timeout command failed. See 10.3 for list of error codes.
Page_Reply_ Timeout	2 octets	*N* = 0xXXXX	Page timeout measured in number of BB slots. Interval length = N * 0.625 ms (1 BB slot) Range for *N*: 0x0001–0xFFFF Time range: 0.625 ms to 40.9 s

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When the Read Page Timeout command has completed, a Command Complete event will be generated.

11.7.3.16 Write Page Timeout command

This command will write the value for the Page_Reply_Timeout configuration parameter. The Page_Reply_Timeout configuration parameter defines the maximum time the local LM will wait for a BB page response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the page at BB level, the connection attempt will be considered to have failed. See Table 254, Table 255, and Table 256.

Table 254—Write Page Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Write_Page_Timeout	0x0018	Page_Reply_Timeout	Status

Table 255—Write Page Timeout command parameters

Parameter	Size	Value	Parameter description
Page_Reply_ Timeout	2 octets	0	Illegal page timeout. Must be larger than 0.
		N = 0xXXXX	Page timeout measured in number of BB slots. Interval length = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0001–0xFFFF Time range: 0.625 ms to 40.9 s Default: *N* = 0x2000 Time = 5.12 s Mandatory range for controller: 0x0016–0xFFFF

Table 256—Write page timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Page Timeout command succeeded.
		0x01–0xFF	Write Page Timeout command failed. See 10.3 for list of error codes.

When the Write Page Timeout command has completed, a Command Complete event will be generated.

11.7.3.17 Read Scan Enable command

This command will read the value for the Scan_Enable parameter configuration parameter. See Table 257 and Table 258. See also 11.6.1.

Table 257—Read Scan Enable command

Command	OCF	Command parameters	Return parameters
HCI_Read_Scan_Enable	0x0019	None	Status, Scan_Enable

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 258—Read Scan Enable command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Scan Enable command succeeded.
		0x01–0xFF	Read Scan Enable command failed. See 10.3 for list of error codes.
Scan_Enable	1 octet	0x00	No scans enabled.
		0x01	Inquiry scan enabled. Page scan disabled.
		0x02	Inquiry scan disabled. Page scan enabled.
		0x03	Inquiry scan enabled. Page scan enabled.
		0x04-0xFF	Reserved.

When the Read Scan Enable command has completed, a Command Complete event will be generated.

11.7.3.18 Write Scan Enable command

This command will write the value for the Scan_Enable configuration parameter. See Table 259, Table 260, and Table 261. See also 11.6.1.

Table 259—Write Scan Enable command

Command	OCF	Command parameters	Return parameters
HCI_Write_Scan_Enable	0x001A	Scan_Enable	Status

Table 260—Write Scan Enable command parameter

Parameter	Size	Value	Parameter description
Scan_Enable	1 octet	0x00	No scans enabled. (default)
		0x01	Inquiry scan enabled. Page scan disabled.
		0x02	Inquiry scan disabled. Page scan enabled.
		0x03	Inquiry scan enabled. Page scan enabled.
		0x04-0xFF	Reserved.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 261—Write Scan Enable command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Scan Enable command succeeded.
		0x01–0xFF	Write Scan Enable command failed. See 10.3 for list of error codes.

When the Write Scan Enable command has completed, a Command Complete event will be generated.

11.7.3.19 Read Page Scan Activity command

This command will read the value for Page_Scan_Interval and Page_Scan_Window configuration parameters. See 11.6.8 and 11.6.9.

Page scan is performed only when page scan is enabled (see 11.6.1, 11.7.3.17, and 11.7.3.18).

A changed page scan interval could change the local page scan repetition (SR) mode [see Table 17 (in 8.6.5.1.4), Table 24 (in 8.8.3.1), and Table 25 (in 8.8.3.2)]. See Table 262 and Table 263.

Table 262—Read Page Scan Activity command

Command	OCF	Command parameters	Return parameters
HCI_Read_Page_Scan_ Activity	0x001B	None	Status, Page_Scan_Interval, Page_Scan_Window

Table 263—Read Page Scan Activity command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Page Scan Activity command succeeded.
		0x01–0xFF	Read Page Scan Activity command failed. See 10.3 for list of error codes.
Page_Scan_Interval	2 octets	*N* = 0xXXXX	Size: 2 octets Range: 0x0012–0x1000 Time = *N* * 0.625 ms Range: 11.25–2560 ms; only even values are valid
Page_Scan_Window	2 octets	*N* = 0xXXXX	Size: 2 octets Range: 0x0011–0x1000 Time = *N* * 0.625 ms Range: 10.625–2560 ms

When the Read Page Scan Activity command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.20 Write Page Scan Activity command

This command will write the values for the Page_Scan_Interval and Page_Scan_Window configuration parameters. The page scan window shall be less than or equal to the page scan interval. See Table 264, Table 265, and Table 266. See also 11.6.8 and 11.6.9.

Table 264—Write Page Scan Activity command

Command	OCF	Command parameters	Return parameters
HCI_Write_Page_Scan_ Activity	0x001C	Page_Scan_Interval, Page_Scan_Window	Status

Table 265—Write Page Scan Activity command parameters

Parameter	Size	Value	Parameter description
Page_Scan_Interval	2 octets		See 11.6.8
Page_Scan_Window	2 octets		See 11.6.9

Table 266—Write Page Scan Activity command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Page Scan Activity command succeeded.
		0x01–0xFF	Write Page Scan Activity command failed. See 10.3 for list of error codes.

Page scan is performed only when page scan is enabled (see 11.6.1, 11.7.3.17, and 11.7.3.18). A changed page scan interval could change the local page SR mode (see 8.8.3.1).

When the Write Page Scan Activity command has completed, a Command Complete event will be generated.

11.7.3.21 Read Inquiry Scan Activity command

This command will read the value for Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameters. See Table 267 and Table 268. See also 11.6.2 and 11.6.3.

Table 267—Read Inquiry Scan Activity command

Command	OCF	Command parameters	Return parameters
HCI_Read_ Inquiry_Scan_Activity	0x001D	None	Status, Inquiry_Scan_Interval, Inquiry_Scan_Window

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 268—Read Inquiry Scan Activity command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Inquiry Scan Activity command succeeded.
		0x01–0xFF	Read Inquiry Scan Activity command failed. See 10.3 for list of error codes.
Inquiry_Scan_ Interval	2 octets	*N* = 0xXXXX	Range: 0x0012–0x1000 Time = *N* * 0.625 ms Range: 11.25–2560 ms; only even values are valid
Inquiry_Scan_ Window	2 octets	*N* = 0xXXXX	Range: 0x0011–0x1000 Time = *N* * 0.625 ms Range: 10.625–2560 ms

Inquiry scan is performed only when inquiry scan is enabled (see 11.6.1, 11.7.3.17, and 11.7.3.18).

When the Read Inquiry Scan Activity command has completed, a Command Complete event will be generated.

11.7.3.22 Write Inquiry Scan Activity command

This command will write the values for the Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameters. The inquiry scan window shall be less than or equal to the inquiry scan interval. See Table 269, Table 270, and Table 271. See also 11.6.2 and 11.6.3.

Table 269—Write Inquiry Scan Activity command

Command	OCF	Command parameters	Return parameters
HCI_Write_Inquiry_ Scan_Activity	0x001E	Inquiry_Scan_Interval, Inquiry_Scan_Window	Status

Table 270—Write Inquiry Scan Activity command parameters

Parameter	Size	Value	Parameter description
Inquiry_Scan_Interval	2 octets		See 11.6.2.
Inquiry_Scan_Window	2 octets		See 11.6.3.

Table 271—Write Inquiry Scan Activity command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Inquiry Scan Activity command succeeded.
		0x01–0xFF	Write Inquiry Scan Activity command failed. See 10.3 for list of error codes.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Inquiry scan is performed only when inquiry scan is enabled (see 11.6.1, 11.7.3.17, and 11.7.3.18).

When the Write Inquiry Scan Activity command has completed, a Command Complete event will be generated.

11.7.3.23 Read Authentication Enable command

This command will read the value for the Authentication_Enable configuration parameter. See Table 272 and Table 273. See also 11.6.15.

Table 272—Read Authentication Enable command

Command	OCF	Command parameters	Return parameters
HCI_Read_ Authentication_Enable	0x001F	None	Status, Authentication_Enable

Table 273—Read Authentication Enable command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Authentication Enable command succeeded.
		0x01–0xFF	Read Authentication Enable command failed. See 10.3 for list of error codes.
Authentication_ Enable	1 octet	0x00	Authentication not required.
		0x01	Authentication required for all connections.
		0x02–0xFF	Reserved.

When the Read Authentication Enable command has completed, a Command Complete event will be generated.

11.7.3.24 Write Authentication Enable command

This command will write the value for the Authentication_Enable configuration parameter. See Table 274, Table 275, and Table 276. See also 11.6.15.

Table 274—Write Authentication Enable command

Command	OCF	Command parameters	Return parameters
HCI_Write_ Authentication_Enable	0x0020	Authentication_Enable	Status

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 275—Write Authentication Enable command parameters

Parameter	Size	Value	Parameter description
Authentication_ Enable	1 octet	0x00	Authentication not required. (default)
		0x01	Authentication required for all connections.
		0x02–0xFF	Reserved.

Table 276—Write Authentication Enable command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Authentication Enable command succeeded.
		0x01–0xFF	Write Authentication Enable command failed. See 10.3 for list of error codes.

When the Write Authentication Enable command has completed, a Command Complete event will be generated.

11.7.3.25 Read Encryption Mode command

This command will read the value for the Encryption_Mode configuration parameter. See Table 277 and Table 278. See also 11.6.16.

Table 277—Read Encryption Mode command

Command	OCF	Command parameters	Return parameters
HCI_Read_Encryption_Mode	0x0021	None	Status, Encryption_Mode

Table 278—Read Encryption Mode command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Encryption Mode command succeeded.
		0x01–0xFF	Read Encryption Mode command failed. See 10.3 for list of error codes.
Encryption_Mode	1 octet		See 11.6.16.

When the Read Encryption Mode command has completed, a Command Complete event will be generated.

11.7.3.26 Write Encryption Mode command

This command will write the value for the Encryption_Mode configuration parameter. See Table 279, Table 280, and Table 281. See also 11.6.16.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 279—Write Encryption Mode command

Command	OCF	Command parameters	Return parameters
HCI_Write_Encryption_Mode	0x0022	Encryption_Mode	Status

Table 280—Write encryption mode command parameter

Parameter	Size	Value	Parameter description
Encryption_Mode	1 octet		See 11.6.16.

Table 281—Write encryption mode command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Encryption Mode command succeeded.
		0x01–0xFF	Write Encryption Mode command failed. See 10.3 for list of error codes.

When the Write Encryption Mode command has completed, a Command Complete event will be generated.

11.7.3.27 Read Class Of Device command

This command will read the value for the Class_of_Device parameter. See Table 282 and Table 283. See also 11.6.25.

Table 282—Read Class Of Device command

Command	OCF	Command parameters	Return parameters
HCI_Read_Class_of_Device	0x0023	None	Status, Class_of_Device

Table 283—Read Class Of Device command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Class Of Device command succeeded.
		0x01–0xFF	Read Class Of Device command failed. See 10.3 for list of error codes.
Class_of_Device	3 octets	0xXXXXXX	Class of the device.

When the Read Class Of Device command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.28 Write Class Of Device command

This command will write the value for the Class_of_Device parameter. See Table 284, Table 285, and Table 286. See also 11.6.25.

Table 284—Write Class Of Device command

Command	OCF	Command parameters	Return parameters
HCI_Write_Class_of_Device	0x0024	Class_of_Device	Status

Table 285—Write Class Of Device command parameter

Parameter	Size	Value	Parameter description
Class_of_Device	3 octets	0xXXXXXX	Class of Device for the device.

Table 286—Write Class Of Device command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Class Of Device command succeeded.
		0x01–0xFF	Write Class Of Device command failed. See 10.3 for list of error codes.

When the Write Class Of Device command has completed, a Command Complete event will be generated.

11.7.3.29 Read Voice Setting command

This command will read the values for the Voice_Setting configuration parameter. See Table 287 and Table 288. See also 11.6.12.

Table 287—Read Voice Setting command

Command	OCF	Command parameters	Return parameters
HCI_Read_Voice_Setting	0x0025	None	Status, Voice_Setting

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 288—Read Voice Setting command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Voice Setting command succeeded.
		0x01–0xFF	Read Voice Setting command failed. See 10.3 for list of error codes.
Voice_Setting	2 octets		See 11.6.12.

When the Read Voice Setting command has completed, a Command Complete event will be generated.

11.7.3.30 Write Voice Setting command

This command will write the values for the Voice_Setting configuration parameter.See Table 289, Table 290, and Table 291. See also 11.6.12.

Table 289—Write Voice Setting command

Command	OCF	Command parameters	Return parameters
HCI_Write_Voice_Setting	0x0026	Voice_Setting	Status

Table 290—Write Voice Setting command parameters

Parameter	Size	Value	Parameter description
Voice_Setting	2 octets		See 11.6.12.

Table 291—Write Voice Setting command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Voice Setting command succeeded.
		0x01–0xFF	Write Voice Setting command failed. See 10.3 for list of error codes.

When the Write Voice Setting command has completed, a Command Complete event will be generated.

11.7.3.31 Read Automatic Flush Timeout command

This command will read the value for the Flush_Timeout parameter for the specified connection handle. See Table 292, Table 293, and Table 294. See also 11.6.20.

When the Read Automatic Flush Timeout command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 292—Read Automatic Flush Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Read_Automatic_Flush_ Timeout	0x0027	Connection_Handle	Status, Connection_Handle, Flush_Timeout

Table 293—Read Automatic Flush Timeout command parameter

Parameter	Size	Value	Parameter description
Connection _Handle	2 octets	0xXXXX	Specifies which connection handle's flush timeout to read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 294—Read automatic flush timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Automatic Flush Timeout command succeeded.
		0x01–0xFF	Read Automatic Flush Timeout command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's flush timeout has been read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Flush_Timeout	2 octets	0	Timeout = ∞; no automatic flush
		N = 0xXXXX	Flush timeout = *N* * 0.625 ms Size: 11 bits Range: 0x0001–0x07FF

11.7.3.32 Write Automatic Flush Timeout command

This command will write the value for the Flush_Timeout parameter for the specified connection handle. See Table 295, Table 296, and Table 297. See also 11.6.20.

Table 295—Write Automatic Flush Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Write_Automatic_Flush_ Timeout	0x0028	Connection_Handle, Flush_Timeout	Status, Connection_Handle

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 296—Write Automatic Flush Timeout command parameters

Parameter	Size	Value	Parameter description
Connection_Handle	2 octets	0xXXXX	Specifies to which connection handle's flush timeout to write. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Flush_Timeout	2 octets	0	Timeout = ∞; no automatic flush. (default)
		N = 0xXXXX	Flush timeout = *N* * 0.625 ms Size: 11 bits Range: 0x0001–0x07FF Mandatory range for controller: 0x0002–0x07FF

Table 297—Write Automatic Flush Timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Automatic Flush Timeout command succeeded.
		0x01–0xFF	Write Automatic Flush Timeout command failed. See 10.3 for list of error codes.
Connection_Handle	2 octets	0xXXXX	Specifies which connection handle's flush timeout has been written. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the Write Automatic Flush Timeout command has completed, a Command Complete event will be generated.

11.7.3.33 Read Num Broadcast Retransmissions command

This command will read the value of a device's Number_of_Broadcast_Retransmissions parameter. See Table 298 and Table 299. See also 11.6.21

Table 298—Read Num Broadcast Retransmissions command

Command	OCF	Command parameters	Return parameters
HCI_Read_Num_Broadcast_ Retransmissions	0x0029	None	Status, Num_Broadcast_Retransmissions

Table 299—Read Num Broadcast Retransmissions command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Num Broadcast Retransmissions command succeeded.
		0x01–0xFF	Read Num Broadcast Retransmissions command failed. See 10.3 for list of error codes.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 299—Read Num Broadcast Retransmissions command return parameters

Num_Broadcast_ Retransmissions	1 octet		See 11.6.21.

When the Read Num Broadcast Retransmission command has completed, a Command Complete event will be generated.

11.7.3.34 Write Num Broadcast Retransmissions command

This command will write the value of a device's Number_of_Broadcast_Retransmissions parameter. See Table 300, Table 301, and Table 302. See also 11.6.21.

Table 300—Write Num Broadcast Retransmissions command

Command	OCF	Command parameters	Return parameters
HCI_Write_Num_Broadcast_ Retransmissions	0x002A	Number_of_Broadcast_ Retransmissions	Status

Table 301—Write Num Broadcast Retransmissions command parameters

Parameter	Size	Value	Parameter description
Number_of_ Broadcast_ Retransmissions	1 octet		See 11.6.21.

Table 302—Write Num Broadcast Retransmissions command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Num Broadcast Retransmissions command succeeded.
		0x01–0xFF	Write Num Broadcast Retransmissions command failed. See 10.3 for list of error codes.

When the Write Num Broadcast Retransmissions command has completed, a Command Complete event will be generated.

11.7.3.35 Read HOLD Mode Activity command

This command will read the value for the Hold_Mode_Activity parameter. See Table 303 and Table 304. See also 11.6.18.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 303—Read HOLD Mode Activity command

Command	OCF	Command parameters	Return parameters
HCI_Read_Hold_Mode_Activity	0x002B	None	Status, Hold_Mode_Activity

Table 304—Read HOLD mode activity command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Hold Mode Activity command succeeded.
		0x01–0xFF	Read Hold Mode Activity command failed. See 10.3 for list of error codes.
Hold_Mode_ Activity	1 octet	0x00	Maintain current power state.
		0x01	Suspend page scan.
		0x02	Suspend inquiry scan.
		0x04	Suspend periodic inquiries.
		0x08–0xFF	Reserved for future use.

When the Read Hold Mode Activity command has completed, a Command Complete event will be generated.

11.7.3.36 Write HOLD Mode Activity command

This command will write the value for the Hold_Mode_Activity parameter. See Table 305, Table 306, and Table 307. See also 11.6.18.

Table 305—Write HOLD Mode Activity command

Command	OCF	Command parameters	Return parameters
HCI_Write_Hold_Mode_Activity	0x002C	Hold_Mode_Activity	Status

Table 306—Write HOLD Mode Activity command parameter

Parameter	Size	Value	Parameter description
Hold_Mode_ Activity	1 octet	0x00	Maintain current power state. **(**default**)**
		0x01	Suspend page scan.
		0x02	Suspend inquiry scan.
		0x04	Suspend periodic inquiries.
		0x08–0xFF	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 307—Write HOLD Mode Activity command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Hold Mode Activity command succeeded.
		0x01–0xFF	Write Hold Mode Activity command failed. See 10.3 for list of error codes.

When the Write Hold Mode Activity command has completed, a Command Complete event will be generated.

11.7.3.37 Read Transmit Power Level command

This command will read the values for the Transmit_Power_Level parameter for the specified connection handle. The connection handle must be a connection handle for an ACL connection. See Table 308, Table 309, and Table 310.

Table 308—Read Transmit Power Level command

Command	OCF	Command parameters	Return parameters
HCI_Read_Transmit_ Power_Level	0x002D	Connection_Handle, Type	Status, Connection_Handle, Transmit_Power_Level

Table 309—Read Transmit Power Level command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's transmit power level setting to read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Type	1 octet	0x00	Read current transmit power level.
		0x01	Read maximum transmit power level.
		0x02–0xFF	Reserved.

Table 310—Read Transmit Power Level Command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Transmit Power Level command succeeded.
		0x01–0xFF	Read Transmit Power Level command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's transmit power level setting is returned. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 310—Read Transmit Power Level Command return parameters *(continued)*

Parameter	Size	Value	Parameter description
Transmit_Power_ Level	1 octet	N = 0xXX	Size: 1 octet (signed integer) Range: $-30 \le N \le 20$ Units: dBm

When the Read Transmit Power Level command has completed, a Command Complete event will be generated.

11.7.3.38 Read Synchronous Flow Control Enable command

The Read Synchronous Flow Control Enable command provides the ability to read the Synchronous_ Flow_Control_Enable parameter setting. See Table 311 and Table 312. See also 11.6.23.

Table 311—Read Synchronous Flow Control Enable command

Command	OCF	Command parameters	Return parameters
HCI_Read_ Synchronous_Flow_ Control_Enable	0x002E	None	Status, Synchronous_Flow_ Control_Enable

Table 312—Read Synchronous Flow Control Enable command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Synchronous Flow Control Enable command succeeded
		0x01–0xFF	Read Synchronous Flow Control Enable command failed. See 10.3 for list of error codes.
Synchronous_ Flow_Control_ Enable	1 octet	0x00	Synchronous flow control is disabled. No Number Of Completed Packets events will be sent from the controller for synchronous connection handles.
		0x01	Synchronous flow control is enabled. Number Of Completed Packets events will be sent from the controller for synchronous connection handles.

The Synchronous_Flow_Control_Enable parameter setting can be changed only if no connections exist.

When the Read Synchronous Flow Control Enable command has completed, a Command Complete event will be generated.

11.7.3.39 Write Synchronous Flow Control Enable command

The Write Synchronous Flow Control Enable command provides the ability to write the Synchronous_Flow_Control_Enable parameter setting. See Table 313, Table 314, and Table 315. See also 11.6.23.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 313—Write Synchronous Flow Control Enable command

Command	OCF	Command parameters	Return parameters
HCI_Write_Synchronous_Flow _Control_Enable	0x002F	Synchronous_Flow_ Control_Enable	Status

Table 314—Write Synchronous Flow Control Enable command parameters

Parameter	Size	Value	Parameter description
Synchronous_Flow_ Control_Enable	1 octet	0x00	Synchronous flow control is disabled. No Number Of Completed Packets events will be sent from the controller for synchronous connection handles. (default)
		0x01	Synchronous flow control is enabled. Number Of Completed Packets events will be sent from the controller for synchronous connection handles.

Table 315—Write Synchronous Flow Control Enable command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Synchronous Flow Control Enable command succeeded
		0x01–0xFF	Write Synchronous Flow Control Enable command failed. See 10.3 for list of error codes.

The Synchronous_Flow_Control_Enable parameter setting can be changed only if no connections exist.

When the Write Synchronous Flow Control Enable command has completed, a Command Complete event will be generated.

11.7.3.40 Set Controller To Host Flow Control command

This command is used by the host to turn flow control on or off for data and/or voice sent in the direction from the controller to the host. If flow control is turned off, the host should not send the Host Number Of Completed Packets command. That command will be ignored by the controller if it is sent by the host and flow control is off. If flow control is turned on for HCI ACL data packets and off for HCI synchronous data packets, Host Number Of Completed Packets commands sent by the host should contain only connection handles for ACL connections. If flow control is turned off for HCI ACL data packets and on for HCI synchronous data packets, Host Number Of Completed Packets commands sent by the host should contain only connection handles for synchronous connections. If flow control is turned on for HCI ACL data packets and HCI synchronous data packets, the host will send Host Number Of Completed Packets commands both for ACL connections and synchronous connections.

The Flow_Control_Enable parameter setting shall be changed only if no connections exist. See Table 316, Table 317, and Table 318.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 316—Set Controller To Host Flow Control command

Command	OCF	Command parameters	Return parameters
HCI_Set_Controller_To_ Host_Flow_Control	0x0031	Flow_Control_Enable	Status

Table 317—Set Controller To Host Flow Control command parameter

Parameter	Size	Value	Parameter description
Flow_Control_ Enable	1 octet	0x00	Flow control off in direction from controller to host. (default)
		0x01	Flow control on for HCI ACL data packets and off for HCI synchronous data packets in direction from controller to host.
		0x02	Flow control off for HCI ACL data packets and on for HCI synchronous data packets in direction from controller to host.
		0x03	Flow control on for both HCI ACL data packets and HCI synchronous data packets in direction from controller to host.
		0x04–0xFF	Reserved.

Table 318—Set Controller To Host Flow Control command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Set Controller To Host Flow Control command succeeded.
		0x01–0xFF	Set Controller To Host Flow Control command failed. See 10.3 for list of error codes.

When the Set Controller To Host Flow Control command has completed, a Command Complete event will be generated.

11.7.3.41 Host Buffer Size command

The Host Buffer Size command is used by the host to notify the controller about the maximum size of the data portion of HCI ACL and synchronous data packets sent from the controller to the host. The controller will segment the data to be transmitted from the controller to the host according to these sizes, so that the HCI data packets will contain data with up to these sizes. The Host Buffer Size command also notifies the controller about the total number of HCI ACL and synchronous data packets that can be stored in the data buffers of the host. If flow control from the controller to the host is turned off and the Host Buffer Size command has not been issued by the host, this means that the controller will send HCI data packets to the host with any lengths the controller wants to use, and it is assumed that the data buffer sizes of the host are unlimited. If flow control from the controller to the host is turned on, the Host Buffer Size command must after a power-on or a reset always be sent by the host before the first Host Number Of Completed Packets command is sent.

(The Set Controller To Host Flow Control command is used to turn flow control on or off.) The Host_ACL_Data_Packet_Length command parameter will be used to determine the size of the L2CAP segments contained in ACL data packets, which are transferred from the controller to the host. The Host_Synchronous_Data_Packet_Length command parameter is used to determine the maximum size of

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

HCI synchronous data packets. Both the host and the controller must support command and event packets, where the data portion (excluding header) contained in the packets is 255 octets in size.

The Host_Total_Num_ACL_Data_Packets command parameter contains the total number of HCI ACL data packets that can be stored in the data buffers of the host. The controller will determine how the buffers are to be divided between different connection handles. The Host_Total_Num_Synchronous_Data_Packets command parameter gives the same information for HCI synchronous data packets. See Table 319, Table 320, and Table 321.

Table 319—Host Buffer Size command

Command	OCF	Command parameters	Return parameters
HCI_Host_Buffer_Size	0x0033	Host_ACL_Data_Packet_Length, Host_Synchronous_Data_Packet_Length, Host_Total_Num_ACL_Data_Packets, Host_Total_Num_Synchronous_Data_Packets	Status

Table 320—Host Buffer Size command parameters

Parameter	Size	Value	Parameter description
Host_ACL_Data_ Packet_Length	2 octets	0xXXXX	Maximum length (in octets) of the data portion of each HCI ACL data packet that the host is able to accept.
Host_Synchronous_ Data_Packet_Length	1 octet	0xXX	Maximum length (in octets) of the data portion of each HCI synchronous data packet that the host is able to accept.
Host_Total_Num_ ACL_Data_Packets	2 octets	0xXXXX	Total number of HCI ACL data packets that can be stored in the data buffers of the host.
Host_Total_Num_ Synchronous_Data_ Packets	2 octets	0xXXXX	Total number of HCI synchronous data packets that can be stored in the data buffers of the host.

Table 321—Host Buffer Size command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Host Buffer Size command succeeded.
		0x01–0xFF	Host Buffer Size command failed. See 10.3 for list of error codes.

The Host_ACL_Data_Packet_Length and Host_Synchronous_Data_Packet_Length command parameters do not include the length of the HCI data packet header.

When the Host Buffer Size command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.42 Host Number Of Completed Packets command

The Host Number Of Completed Packets command is used by the host to indicate to the controller the number of HCI data packets that have been completed for each connection handle since the previous Host Number Of Completed Packets command was sent to the controller. This means that the corresponding buffer space has been freed in the host. Based on this information, and the Host_Total_Num_ACL_Data_Packets and Host_Total_Num_Synchronous_Data_Packets command parameters of the Host Buffer Size command, the controller can determine for which connection handles the following HCI data packets should be sent to the host. The command should be issued by the host only if flow control in the direction from the controller to the host is on and there is at least one connection or if the controller is in local loopback mode. Otherwise, the command will be ignored by the controller. When the host has completed one or more HCI data packet(s), it shall send a Host Number Of Completed Packets command to the controller until it finally reports that all pending HCI data packets have been completed. The frequency at which this command is sent is manufacturer specific.

(The Set Controller To Host Flow Control command is used to turn flow control on or off.) If flow control from the controller to the host is turned on, the Host Buffer Size command must after a power-on or a reset always be sent by the host before the first Host Number Of Completed Packets command is sent. See Table 322 and Table 323.

Table 322—Host Number Of Completed Packets command

Command	OCF	Command parameters	Return parameters
HCI_Host_Number_Of_ Completed_Packets	0x0035	Number_of_Handles, Connection_Handle[i], Host_Num_of_Completed_Packets[i]	None

Table 323—Host Number Of Completed Packets command parameters

Parameter	Size	Value	Parameter description
Number_of_ Handles	1 octet	0xXX	The number of connection handles and Host_Num_of_ Completed_Packets parameter pairs contained in this command. Range: 0–255
Connection_ Handle[i]	Number_of_ Handles * 2 octets	0xXXXX	Connection handle Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Host_Num_ of_Completed _Packets[i]	Number_of_ Handles * 2 octets	*N* = 0xXXXX	The number of HCI data packets that have been completed for the associated connection handle since the previous time the event was returned. Range for *N*: 0x0000–0xFFFF

The Host Number Of Completed Packets command is a special command in the sense that no event is normally generated after the command has completed. The command may be sent at any time by the host when there is at least one connection or if the controller is in local loopback mode independent of other commands. The normal flow control for commands is not used for the Host Number Of Completed Packets command.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Normally, no event is generated after the Host Number Of Completed Packets command has completed. However, if the Host Number Of Completed Packets command contains one or more invalid parameters, the controller will return a Command Complete event with a failure status indicating the error code *invalid HCI command parameters*. The host may send the Host Number Of Completed Packets command at any time when there is at least one connection or if the controller is in local loopback mode. The normal flow control for commands is not used for this command.

11.7.3.43 Read Link Supervision Timeout command

This command will read the value for the Link_Supervision_Timeout parameter for the device. See Table 324, Table 325, and Table 326.

Table 324—Read Link Supervision Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Read_Link_Supervision_Timeout	0x0036	Connection_Handle	Status, Connection_Handle, Link_Supervision_Timeout

Table 325—Read Link Supervision Timeout command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's link supervision timeout value is to be read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 326—Read Link Supervision Timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Link Supervision Timeout command succeeded.
		0x01–0xFF	Read Link Supervision Timeout command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's link supervision timeout value was read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Link_Supervision_ Timeout	2 octets	0x0000	No link supervision timeout.
		N = 0xXXXX	Measured in number of BB slots Link supervision timeout = *N* * 0.625 ms (1 BB slot) Range for N: 0x0001–0xFFFF Time range: 0.625 ms to 40.9 s

The connection handle used for this command must be the ACL connection to the appropriate device. See 11.6.22.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

When the Read Link Supervision Timeout command has completed, a Command Complete event will be generated.

11.7.3.44 Write Link Supervision Timeout command

This command will write the value for the Link_Supervision_Timeout parameter for the device. See Table 327, Table 328, and Table 329.

Table 327—Write Link Supervision Timeout command

Command	OCF	Command parameters	Return parameters
HCI_Write_Link_Supervision_ Timeout	0x0037	Connection_Handle, Link_Supervision_Timeout	Status, Connection_Handle

Table 328—Write Link Supervision Timeout command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's link supervision timeout value is to be written. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Link_ Supervision_ Timeout	2 octets	0x0000	No link supervision timeout.
		N = 0xXXXX	Measured in number of BB slots Link supervision timeout = N*0.625 ms (1 BB slot) Range for *N*: 0x0001–0xFFFF Time range: 0.625 ms to 40.9 s Default: *N* = 0x7D00 Link supervision timeout = 20 s Mandatory range for controller: 0x0190–0xFFFF; plus 0 for infinite timeout

Table 329—Write Link Supervision Timeout command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Link Supervision Timeout command succeeded.
		0x01–0xFF	Write Link Supervision Timeout command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's link supervision timeout value was written. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

The connection handle used for this command must be the ACL connection to the appropriate device. This command will set the link supervision timeout values for other synchronous connection handles to that device. See 11.6.22.

When the Write Link Supervision Timeout command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.45 Read Number Of Supported IAC command

This command will read the value for the number of IACs for which the local device can simultaneously listen during an inquiry scan. All devices are required to support at least one IAC, the GIAC. Some devices support additional IACs. See Table 330 and Table 331.

Table 330—Read Number Of Supported IAC command

Command	OCF	Command parameters	Return parameters
HCI_Read_Number_Of_Supported_IAC	0x0038	None	Status, Num_Support_IAC

Table 331—Read Number Of Supported IAC command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Number Of Supported IAC command succeeded.
		0x01–0xFF	Read Number Of Supported IAC command failed. See 10.3 for list of error codes.
Num_Support_IAC	1 octet	0xXX	Specifies the number of supported IAC for which the local device can simultaneously listen during an inquiry scan. Range: 0x01–0x40

When the Read Number Of Supported IAC command has completed, a Command Complete event will be generated.

11.7.3.46 Read Current IAC LAP command

This command reads the LAP(s) used to create the IACs for which the local device is simultaneously scanning during inquiry scans. All IEEE 802.15.1-2005 devices are required to support at least one IAC, the GIAC. Some devices support additional IACs. See Table 332 and Table 333.

Table 332—Read Current IAC LAP command

Command	OCF	Command parameters	Return parameters
HCI_Read_Current_IAC_LAP	0x0039	None	Status, Num_Current_IAC, IAC_LAP[i]

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 333—Read Current IAC LAP command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Current IAC LAP command succeeded.
		0x01–0xFF	Read Current IAC LAP command failed. See 10.3 for list of error codes.
Num_ Current_IAC	1 octet	0xXX	Specifies the number of IACs for which the local device is currently and simultaneously listening during an inquiry scan. Range: 0x01–0x40
IAC_LAP[i]	3 octets * Num_Current _IAC	0xXXXXXX	LAPs used to create the IAC for which the local device is currently and simultaneously listening during an inquiry scan. Range: 0x9E8B00–0x9E8B3F

When the Read Current IAC LAP command has completed, a Command Complete event will be generated.

11.7.3.47 Write Current IAC LAP command

This command writes the LAP(s) used to create the IACs for which the local device is simultaneously scanning during inquiry scans. All devices are required to support at least one IAC, the GIAC. Some devices support additional IACs.

This command shall clear any existing IACs and stores the number of current IACs and the IAC LAPs in to the controller. If the number of current IACs is greater than number of supported IACs (see 11.7.3.45), then only the first number of supported IACs shall be stored in the controller, and a Command Complete event with error code *success* (0x00) shall be generated. See Table 334, Table 335, and Table 336.

Table 334—Write Current IAC LAP command

Command	OCF	Command parameters	Return parameters
HCI_Write_Current_IAC_LAP	0x003A	Num_Current_IAC, IAC_LAP[i]	Status

Table 335—Write Current IAC LAP command parameters

Parameter	Size	Value	Parameter description
Num_Current_IAC	1 octet	0xXX	Specifies the number of IACs for which the local device is currently and simultaneously listening during an inquiry scan. Range: 0x01–0x40
IAC_LAP[i]	3 octets * Num_Current_ IAC	0xXXXXXX	LAP(s) used to create IAC for which the local device is currently and simultaneously listening during an inquiry scan. Range: 0x9E8B00–0x9E8B3F. The GIAC is the default IAC to be used. If additional IACs are supported, additional default IAC will be determined by the manufacturer.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 336—Write Current IAC LAP command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Current IAC LAP command succeeded.
		0x01–0xFF	Write Current IAC LAP command failed. See 10.3 for list of error codes.

When the Write Current IAC LAP command has completed, a Command Complete event will be generated.

11.7.3.48 Read Page Scan Period Mode command

This command is used to read the mandatory Page_Scan_Period_Mode configuration parameter of the local device. See Table 337 and Table 338. See also 11.6.10.

Table 337—Read Page Scan Period Mode command

Command	OCF	Command Parameters	Return parameters
HCI_Read_Page_Scan_Period_ Mode	0x003B	None	Status, Page_Scan_Period_Mode

Table 338—Read Page Scan Period Mode command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Page Scan Period Mode command succeeded.
		0x01–0xFF	Read Page Scan Period Mode command failed. See 10.3 for list of error codes.
Page_Scan_ Period_Mode	1 octet	0x00	P0
		0x01	P1
		0x02	P2
		0x03–0xFF	Reserved.

When the Read Page Scan Period Mode command has completed, a Command Complete event will be generated.

11.7.3.49 Write Page Scan Period Mode command

This command is used to write the mandatory Page_Scan_Period_Mode configuration parameter of the local device. See Table 339, Table 340, and Table 341. See also 11.6.10.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 339—Write Page Scan Period Mode command

Command	OCF	Command parameters	Return parameters
HCI_Write_Page_Scan_Period_Mode	0x003C	Page_Scan_Period_Mode	Status

Table 340—Write Page Scan Period Mode command parameters

Parameter	Size	Value	Parameter description
Page_Scan_Period_Mode	1 octet	0x00	P0. (default)
		0x01	P1.
		0x02	P2.
		0x03–0xFF	Reserved.

Table 341—Write Page Scan Period Mode command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Page Scan Period Mode command succeeded.
		0x01–0xFF	Write Page Scan Period Mode command failed. See 10.3 for list of error codes.

When the Write Page Scan Period Mode command has completed, a command Complete event will be generated.

11.7.3.50 Set AFH Host Channel Classification command

The Set AFH Host Channel Classification command allows the host to specify a channel classification based on its "local information." This classification persists until overwritten with a subsequent HCI Set AFH Host Channel Classification command or until the controller is reset.

This command shall be supported by a device that declares support for any of the AFH-capable master, AFH classification slave, or AFH classification master feature (see Table 34 in 9.2.1).

If this command is used, then updates should be sent within 10 s after the host learns that the channel classification has changed. The interval between two successive commands sent shall be at least 1 s. See Table 342, Table 343, and Table 344. :

Table 342—Set AFH Host Channel Classification command

Command	OCF	Command parameters	Return parameters
Set_AFH_Host_Channel_Classification	0x003F	AFH_Host_Channel_Classification	Status

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 343—Set AFH host channel classification command parameter

Parameter	Size	Value	Parameter description
AFH_Host_Channel_Classification	10 octets	0xXXXXX XXXXXX XXXXXX XXX	This parameter contains 79 one-bit field. The nth such field (in the range 0 to 78) contains the value for channel n: Channel n is bad = 0 Channel n is unknown = 1 The MSB is reserved and shall be set to 0. At least N_{min} channels shall be marked as unknown. (See 8.2.3.1.)

Table 344—Set AFH Host Channel Classification command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Set AFH Host Channel Classification command succeeded.
		0x01–0xFF	Set AFH Host Channel Classification command failed. See 10.3 for list of error codes.

When the Set AFH Host Channel Classification command has completed, a Command Complete event will be generated.

11.7.3.51 Read Inquiry Scan Type command

This command is used to read the Inquiry_Scan_Type configuration parameter of the local device. See Table 345 and Table 346. See also 11.6.4. For details on the **inquiry scan** substate, see 8.8.4.1.

Table 345—Read Inquiry Scan Type command

Command	OCF	Command parameters	Return parameters
HCI_Read_Inquiry_Scan_Type	0x0042	None	Status, Inquiry_Scan_Type

Table 346—Read Inquiry Scan Type command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Inquiry Scan Type command succeeded.
		0x01–0xFF	Read Inquiry Scan Type command failed. See 10.3 for list of error codes.
Inquiry_Scan_Type	1 octet	0x00	Mandatory: standard scan. (default)
		0x01	Optional: interlaced scan.
		0x02–0xFF	Reserved.

When the Read Inquiry Scan Type command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.52 Write Inquiry Scan Type command

This command is used to write the Inquiry_Scan_Type configuration parameter of the local device. See Table 347, Table 348, and Table 349. See also 11.6.4. For details on the **inquiry scan** substate, see 8.8.4.1.

Table 347—Write Inquiry Scan Type command

Command	OCF	Command parameters	Return parameters
HCI_Write_Inquiry_Scan_Type	0x0043	Scan_Type	Status

Table 348—Write Inquiry Scan Type command parameter

Parameter	Size	Value	Parameter description
Scan_Type	1 octet	0x00	Mandatory: standard scan. (default)
		0x01	Optional: interlaced scan.
		0x02–0xFF	Reserved.

Table 349—Write Inquiry Scan Type command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Inquiry Scan Type command succeeded
		0x01–0xFF	Write Inquiry Scan Type command failed. See 10.3 for list of error codes.

When the Write Inquiry Scan Type command has completed, a Command Complete event will be generated.

11.7.3.53 Read Inquiry Mode command

This command is used to read the Inquiry_Mode configuration parameter of the local device. See Table 350 and Table 351. See also 11.6.5.

Table 350—Read Inquiry Mode command

Command	OCF	Command parameters	Return parameters
HCI_Read_Inquiry_Mode	0x0044	None	Status, Inquiry_Mode

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 351—Read Inquiry Mode command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Inquiry Mode command succeeded.
		0x01–0xFF	Read Inquiry Mode command failed. See 10.3 for list of error codes.
Inquiry_Mode	1 octet	0x00	Standard Inquiry Result event format.
		0x01	Inquiry Result with RSSI event format.
		0x02–0xFF	Reserved.

When the Read Inquiry Mode command has completed, a Command Complete event will be generated.

11.7.3.54 Write Inquiry Mode command

This command is used to write the Inquiry_Mode configuration parameter of the local device. See Table 352, Table 353, and Table 354. See also 11.6.5.

Table 352—Write Inquiry Mode command

Command	OCF	Command parameters	Return parameters
HCI_Write_Inquiry_Mode	0x0045	Inquiry_Mode	Status

Table 353—Write Inquiry Mode command parameters

Parameter	Size	Value	Parameter description
Inquiry_Mode	1 octet	0x00	Standard Inquiry Result event format. (default)
		0x01	Inquiry Result With RSSI event format.
		0x02–0xFF	Reserved.

Table 354—Write Inquiry Mode command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Inquiry Mode command succeeded
		0x01–0xFF	Write Inquiry Mode command failed. See 10.3 for list of error codes.

When the Write Inquiry Mode command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.3.55 Read Page Scan Type command

This command is used to read the Page_Scan_Type configuration parameter of the local device. See Table 355 and Table 356. See also 11.6.11. For details on the **page scan** substate, see 8.8.3.1.

Table 355—Read Page Scan Type command

Command	OCF	Command parameters	Return parameters
HCI_Read_Page_Scan_Type	0x0046	None	Status, Page_Scan_Type

Table 356—Read Page Scan Type command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Page Scan Type command succeeded.
		0x01–0xFF	Read Page Scan Type command failed. See 10.3 for list of error codes.
Page_Scan_ Type	1 octet	0x00	Mandatory: standard scan. (default)
		0x01	Optional: interlaced scan.
		0x02–0xFF	Reserved.

When the Read Page Scan Type command has completed, a Command Complete event will be generated.

11.7.3.56 Write Page Scan Type command

This command is used to write the Page_Scan_Type configuration parameter of the local device. See Table 357, Table 358, and Table 359. See also 11.6.11. For details on **page scan** substate, see 8.8.3.1.

Table 357—Write Page Scan Type command

Command	OCF	Command parameters	Return parameters
HCI_Write_Page_Scan_Type	0x0047	Page_Scan_Type	Status

Table 358—Write Page Scan Type command parameters

Parameter	Size	Value	Parameter description
Page_Scan_ Type	1 octet	0x00	Mandatory: standard scan. (default)
		0x01	Optional: interlaced scan.
		0x02–0xFF	Reserved.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 359—Write Page Scan Type command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Page Scan Type command succeeded.
		0x01–0xFF	Write Page Scan Type command failed. See 10.3 for list of error codes.

When the Write Page Scan Type command has completed, a Command Complete event will be generated.

11.7.3.57 Read AFH Channel Assessment Mode command

The Read AFH Channel Assessment Mode command reads the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the controller's channel assessment scheme is enabled or disabled.

This command shall be supported by a device that declares support for any of the AFH-capable master, AFH classification slave, or AFH classification master feature (see Table 34 in 9.2.1). See Table 360 and Table 361.

Table 360—Read AFH Channel Assessment Mode command

Command	OCF	Command parameters	Return parameters
Read_AFH_Channel_Assessment_Mode	0x0048	None	Status, AFH_Channel_Assessment_Mode

Table 361—Read AFH Channel Assessment Mode command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read AFH Channel Assessment Mode command succeeded.
		0x01–0xFF	Read AFH Channel Assessment Mode command failed. See 10.3 for list of error codes.
AFH_Channel_Assessment_Mode	1 octet	0x00	Controller channel assessment disabled.
		0x01	Controller channel assessment enabled.
		0x02–0xFF	Reserved for future use.

When the Read AFH Channel Assessment Mode command has completed, a Command Complete event will be generated.

11.7.3.58 Write AFH Channel Assessment Mode command

The Write AFH Channel Assessment Mode command writes the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the controller's channel assessment scheme is enabled or disabled.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Disabling channel assessment forces all channels to be unknown in the local classification, but does not affect the AFH reporting mode or support for the Set AFH Host Channel Classification command. A slave in the AFH reporting enabled mode shall continue to send LMP channel classification messages for any changes to the channel classification caused by either this command (altering the AFH channel assessment mode) or HCI Set AFH Host Channel Classification command (providing a new channel classification from the host).

This command shall be supported by a device that declares support for any of the AFH-capable master, AFH classification slave, or AFH classification master feature (see Table 34 in 9.2.1).

If the AFH_Channel_Assessment_Mode parameter is enabled and the controller does not support a channel assessment scheme, other than via the Set AFH Host Channel Classification command, then a Status parameter with the error code *channel classification not supported* should be returned. See 10.3 for list of error codes.

If the controller supports a channel assessment scheme, then the default AFH channel assessment mode is enabled; otherwise, the default is disabled. See Table 362, Table 363, and Table 364. .

Table 362—Write AFH Channel Assessment Mode command

Command	OCF	Command parameters	Return parameters
Write_AFH_Channel_Assessment_Mode	0x0049	AFH_Channel_Assessment_Mode	Status

Table 363—Write AFH Channel Assessment Mode command parameter

Parameter	Size	Value	Parameter description
AFH_Channel_ Assessment_Mode	1 octet	0x00	Controller channel assessment disabled.
		0x01	Controller channel assessment enabled.
		0x02–0xFF	Reserved for future use.

Table 364—Write AFH Channel Assessment Mode command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write AFH Channel Assessment Mode command succeeded.
		0x01–0xFF	Write AFH Channel Assessment Mode command failed. See 10.3 for list of error codes.

When the Write AFH Channel Assessment Mode command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.4 Informational parameters

The informational parameters are fixed by the manufacturer of the hardware. These parameters provide information about the device and the capabilities of the controller, LM, and BB. The host device cannot modify any of these parameters. For informational parameters commands, the OGF is defined as 0x04.

11.7.4.1 Read Local Version Information command

This command will read the values for the version information for the local device.

The HCI_Version parameter defines the version information of the HCI layer. The LMP_Version parameter defines the version of the LMP. The Manufacturer_Name parameter indicates the manufacturer of the local device.

The HCI_Revision and LMP_Subversion parameters are implementation dependent.

See Table 365 and Table 366.

Table 365—Read Local Version Information command

Command	OCF	Command parameters	Return parameters
HCI_Read_Local_Version_Information	0x0001	None	Status, HCI_Version, HCI_Revision, LMP_Version, Manufacturer_Name, LMP_Subversion

Table 366—Read Local Version Information command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Local Version Information command succeeded.
		0x01–0xFF	Read Local Version Information command failed. See 10.3 for list of error codes.
HCI_Version	1 octet		See Bluetooth Assigned Numbers [B1].
HCI_Revision	2 octets	0xXXXX	Revision of the current HCI in the device.
LMP_Version	1 octet	0xXX	Version of the current LMP in the device. See Bluetooth Assigned Numbers.
Manufacturer_ Name	2 octets	0xXXXX	Manufacturer name of the device. See Bluetooth Assigned Numbers.
LMP_ Subversion	2 octets	0xXXXX	Subversion of the current LMP in the device. See Table 68 in 9.4.1 for assigned values (SubVersNr).

When the Read Local Version Information command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.4.2 Read Local Supported Commands command

This command reads the list of HCI commands supported for the local device.

This command will return the Supported_Commands configuration parameter. It is implied that if a command is listed as supported, the feature underlying that command is also supported.

See Table 367 and Table 368. See also 11.6.26 for more information.

Table 367—Read Local Supported Commands command

Command	OCF	Command parameters	Return parameters
HCI_ Read_Local_Supported_Commands	0x0002		Status, Supported_Commands

Table 368—Read Local Supported Commands command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0	Read Local Supported Commands command succeeded.
		0x01–0xff	Read Local Supported Commands command failed. See 10.3 for list of error codes.
Supported_ Commands	64 octets		Bit mask for each HCI command. If a bit is 1, the radio supports the corresponding command and the features required for the command. Unsupported or undefined commands shall be set to 0. See 11.6.26.

When the Read Local Supported Commands command has completed, a Command Complete event will be generated.

11.7.4.3 Read Local Supported Features command

This command requests a list of the supported features for the local device. This command will return a list of the LMP features. See Table 369 and Table 370. For details, see Clause 9.

Table 369—Read Local Supported Features command

Command	OCF	Command parameters	Return parameters
HCI_Read_Local_Supported_Features	0x0003	None	Status, LMP_Features

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 370—Read Local Supported Features command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Local Supported Features command succeeded.
		0x01–0xFF	Read Local Supported Features command failed. See 10.3 for list of error codes.
LMP_ Features	8 octets	0xXXXXX XXXXXX XXXXX	Bit mask list of LMP features. For details, see Clause 9.

When the Read Local Supported Features command has completed, a Command Complete event will be generated.

11.7.4.4 Read Local Extended Features command

The Read Local Extended Features command returns the requested page of the extended LMP features. See Table 371, Table 372, and Table 373.

Table 371—Read Local Extended Features command

Command	OCF	Command parameters	Return parameters
HCI_Read_Local_ Extended_Features	0x0004	Page_Number	Status, Page_Number, Maximum_Page_Number, Extended_LMP_Features

Table 372—Read Local Extended Features command parameter

Parameter	Size	Value	Parameter description
Page_Number	1 octet	0x00	Requests the normal LMP features as returned by Read Local Supported Features command.
		0x01–0xFF	Return the corresponding page of features.

Table 373—Read Local Extended Features command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Local Extended Features command succeeded.
		0x01–0xFF	Read Local Extended Features command failed. See 10.3 for list of error codes.
Page_Number	1 octet	0x00	The normal LMP features as returned by Read Local Supported Features command.
		0x01–0xFF	The page number of the features returned.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 373—Read Local Extended Features command return parameters *(continued)*

Parameter	Size	Value	Parameter description
Maximum_ Page_Number	1 octet	0x00–0xFF	The highest features page number that contains nonzero bits for the local device.
Extended_ LMP_Features	8 octets	0xFFFFFFF FFFFFFFF	Bit map of requested page of LMP features. See Clause 9 for details.

When the controller receives the Read Local Extended Features command, the controller sends the Command Complete command to the host containing the requested information.

11.7.4.5 Read Buffer Size command

The Read Buffer Size command is used to read the maximum size of the data portion of HCI ACL and synchronous data packets sent from the host to the controller. The host will segment the data to be transmitted from the host to the controller according to these sizes, so that the HCI data packets will contain data with up to these sizes. The Read Buffer Size command also returns the total number of HCI ACL and synchronous data packets that can be stored in the data buffers of the controller. The Read Buffer Size command must be issued by the host before it sends any data to the controller.

The HC_ACL_Data_Packet_Length return parameter will be used to determine the size of the L2CAP segments contained in ACL data packets, which are transferred from the host to the controller to be broken up into BB packets by the LM. The HC_Synchronous_Data_Packet_Length return parameter is used to determine the maximum size of HCI synchronous data packets. Both the host and the controller must support command and event packets, where the data portion (excluding header) contained in the packets is 255 octets in size. The HC_Total_Num_ACL_Data_Packets return parameter contains the total number of HCI ACL data packets that can be stored in the data buffers of the controller. The host will determine how the buffers are to be divided between different connection handles. The HC_Total_Num_Synchronous_ Data_Packets return parameter gives the same information, but for HCI synchronous data packets.

See Table 374 and Table 375.

Table 374—Read Buffer Size command

Command	OCF	Command parameters	Return parameters
HCI_Read_Buffer_Size	0x0005	None	Status, HC_ACL_Data_Packet_Length, HC_Synchronous_Data_Packet_Length, HC_Total_Num_ACL_Data_Packets, HC_Total_Num_Synchronous_Data_Packets

Table 375—Read Buffer Size command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Buffer Size command succeeded.
		0x01–0xFF	Read Buffer Size command failed. See 10.3 for list of error codes.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 375—Read Buffer Size command return parameters *(continued)*

Parameter	Size	Value	Parameter description
HC_ACL_Data_ Packet_Length	2 octets	0xXXXX	Maximum length (in octets) of the data portion of each HCI ACL data packet that the controller is able to accept.
HC_Synchronous_ Data_Packet_ Length	1 octet	0xXX	Maximum length (in octets) of the data portion of each HCI synchronous data packet that the controller is able to accept.
HC_Total_Num_ ACL_Data_Packets	2 octets	0xXXXX	Total number of HCI ACL data packets that can be stored in the data buffers of the controller.
HC_Total_Num_ Synchronous_Data_ Packets	2 octets	0xXXXX	Total number of HCI synchronous data packets that can be stored in the data buffers of the controller.

The HC_ACL_Data_Packet_Length and HC_Synchronous_Data_Packet_Length return parameters do not include the length of the HCI data packet header.

When the Read Buffer Size command has completed, a Command Complete event will be generated.

11.7.4.6 Read BD_ADDR command

This command shall read the device address (BD_ADDR). See Clause 8, for details of how BD_ADDR is used. See Table 376 and Table 377.

Table 376—Read BD_ADDR command

Command	OCF	Command parameters	Return parameters
HCI_Read_BD_ADDR	0x0009	None	Status, BD_ADDR

Table 377—Read BD_ADDR command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read BD_ADDR command succeeded.
		0x01–0xFF	Read BD_ADDR command failed. See 10.3 for list of error codes.
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device.

When the Read BD_ADDR command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.5 Status parameters

The controller modifies all status parameters. These parameters provide information about the current state of the controller, LM, and BB. The host device cannot modify any of these parameters other than to reset certain specific parameters. For the status and BB, the OGF is defined as 0x05.

11.7.5.1 Read Failed Contact Counter command

This command will read the value for the Failed_Contact_Counter parameter for a particular connection to another device. The connection handle must be a connection handle for an ACL connection. See Table 378, Table 379, and Table 380. See also 11.6.17.

Table 378—Read Failed Contact Counter command

Command	OCF	Command parameters	Return parameters
HCI_Read_Failed_ Contact_Counter	0x0001	Connection_Handle	Status, Connection_Handle, Failed_Contact_Counter

Table 379—Read Failed Contact Counter command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the failed contact counter should be read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 380—Read Failed Contact Counter command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Failed Contact Counter command succeeded.
		0x01–0xFF	Read Failed Contact Counter command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the failed contact counter has been read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Failed_Contact_ Counter	2 octets	0xXXXX	Number of consecutive failed contacts for a connection corresponding to the connection handle.

When the Read Failed Contact Counter command has completed, a Command Complete event will be generated.

11.7.5.2 Reset Failed Contact Counter command

This command will reset the value for the Failed_Contact_Counter parameter for a particular connection to another device. The connection handle must be a connection handle for an ACL connection. See Table 381, Table 382, and Table 383. See also 11.6.17.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 381—Reset Failed Contact Counter command

Command	OCF	Command parameters	Return parameters
HCI_Reset_Failed_ Contact_Counter	0x0002	Connection_Handle	Status, Connection_Handle

Table 382—Reset Failed Contact Counter command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the failed contact counter should be reset. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 383—Reset Failed Contact Counter command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00 0x01–0xFF	Reset Failed Contact Counter command succeeded. Reset Failed Contact Counter command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the failed contact counter has been reset. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

When the Reset Failed Contact Counter command has completed, a Command Complete event will be generated.

11.7.5.3 Read link quality command

This command will return the value for the Link_Quality parameter for the specified connection handle. The connection handle must be a connection handle for an ACL connection. This command will return a Link_Quality parameter value from 0 to 255, which represents the quality of the link between two devices. The higher the value, the better the link quality is. Each module vendor will determine how to measure the link quality. See Table 384, Table 385, and Table 386.

Table 384—Read Link Quality command

Command	OCF	Command parameters	Return parameters
HCI_Read_Link_Quality	0x0003	Connection_Handle	Status, Connection_Handle, Link_Quality

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 385—Read Link Quality command parameter

Parameter	Size	Value	Parameter description
Connection _Handle	2 octets	0xXXXX	The connection handle for the connection for which link quality parameters are to be read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 386—Read Link Quality command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Link Quality command succeeded.
		0x01–0xFF	Read Link Quality command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the link quality parameter has been read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Link_Quality	1 octet	0xXX	The current quality of the link connection between the local device and remote device specified by the connection handle. Range: 0x00–0xFF The higher the value, the better the link quality is.

When the Read Link Quality command has completed, a Command Complete event will be generated.

11.7.5.4 Read RSSI command

This command will read the value for the difference between the measured RSSI and the limits of the golden receive power range (see 7.4.6) for a connection handle to another device. The connection handle must be a connection handle for an ACL connection. Any positive RSSI value returned by the controller indicates how many decibels the RSSI is above the upper limit; any negative value indicates how many decibels the RSSI is below the lower limit. The value zero indicates that the RSSI is inside the golden receive power range.

How accurate the decibel values will be depends on the hardware. The only requirements for the hardware are that the device is able to tell whether the RSSI is inside, above, or below the golden device power range.

The RSSI measurement compares the received signal power with two threshold levels that define the golden receive power range. The lower threshold level corresponds to a received power between –56 dBm and 6 dB above the actual sensitivity of the receiver. The upper threshold level is 20 dB above the lower threshold level to an accuracy of ± 6 dB. See Table 387, Table 388, and Table 389.

Table 387—Read RSSI command

Command	OCF	Command parameters	Return parameters
HCI_Read_RSSI	0x0005	Connection_Handle	Status, Connection_Handle, RSSI

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 388—Read RSSI command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the RSSI is to be read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 389—Read RSSI command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read RSSI command succeeded.
		0x01–0xFF	Read RSSI command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the RSSI has been read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
RSSI	1 octet	N = 0xXX	Size: 1 octet (signed integer) Range: –128 ≤ N ≤ 127 Units: decibels

When the Read RSSI command has completed, a Command Complete event will be generated.

11.7.5.5 Read AFH Channel Map command

This command will return the values for the AFH_Mode and AFH_Channel_Map parameters for the specified connection handle. The connection handle must be a connection handle for an ACL connection.

The returned values indicate the state of the hop sequence specified by the most recent LMP_Set_AFH message for the specified connection handle, regardless of whether the master has received the BB ACK or whether the AFH instant has passed. This command shall be supported by a device that declares support for either the AFH-capable slave or AFH-capable master feature. See Table 390, Table 391, and Table 392.

Table 390—Read AFH Channel Map command

Command	OCF	Command parameters	Return parameters
Read_AFH_Channel_Map	0x0006	Connection_Handle	Status, Connection_Handle, AFH_Mode, AFH_Channel_Map

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 391—Read AFH Channel Map command parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the channel map is to be read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 392—Read AFH Channel Map Command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read AFH Channel Map command succeeded.
		0x01–0xFF	Read AFH Channel Map command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the channel map is to be read. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
AFH_Mode	1 octet	0x00	AFH disabled.
		0x01	AFH enabled.
		0x02–0xFF	Reserved for future use.
AFH_Channel_ Map	10 octets	0xXXXXX XXXXXX XXXXXX XXX	If AFH mode is AFH enabled, then this parameter contains 79 one-bit fields; otherwise, the contents are reserved. The nth such field (in the range 0 to 78) contains the value for channel *n*: Channel *n* is unused = 0 Channel *n* is used = 1 Range: 0x00000000000000000000– 0x7FFFFFFFFFFFFFFFFFFF (0x80000000000000000000– 0xFFFFFFFFFFFFFFFFFFFF Reserved for future use)

When the Read AFH Channel Map command has completed, a Command Complete event will be generated.

11.7.5.6 Read Clock command

This command will read the estimate of the value of the CLK. See Table 393, Table 394, and Table 395.

Table 393—Read Clock command

Command	OCF	Command parameters	Return parameters
HCI_Read_Clock	0x0007	Which_Clock Connection_Handle	Status, Connection_Handle, Clock, Accuracy

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 394—Read Clock command parameters

Parameter	Size	Value	Parameter description
Which_Clock	1 octet	0x00	Local clock (connection handle does not have to be a valid handle).
		0x01	Piconet clock (connection handle shall be a valid ACL Handle).
		0x02–0xFF	Reserved.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify which connection to be used for reading the CLK. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

Table 395—Read Clock command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read CLOCK command succeeded.
		0x01–0xFF	Read CLOCK command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	The connection handle for the connection for which the CLK has been read. If the Which_Clock parameter was 0, then the connection handle shall be set to 0 and ignored upon receipt. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Clock	4 octets	0xXXXXX XXX	Clock of the device requested.
Accuracy	2 octets	0xXX	± maximum clock error. Value of 0xFFFF means *unknown*. Accuracy = ± *N* * 0.3125 ms (1 clock cycle) Range for *N*: 0x00–0xFFFE, 0xFFFF Time range for *N*: 0–20479.375 s

If the Which_Clock parameter value is 0, then the connection handle shall be ignored, the CLKE shall be returned, and the accuracy parameter shall be set to 0.

If the Which_Clock paramter value is 1, then the connection handle must be a valid ACL connection handle. If the current role of this ACL connection is master, then the CLKE of this device shall be returned. If the current role is slave, then an estimate of the CLKN of the remote master and the accuracy of this value shall be returned.

The accuracy reflects the clock drift that might have occurred since the slave last received a valid transmission from the master.

The clock has a minimum accuracy of 250 ppm, or about 22 s, drift in one day.

See 8.1.1 for more information about the clock.

When the Read Clock command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.6 Testing commands

The Testing commands are used to provide the ability to test various functionalities of the hardware. These commands provide the ability to arrange various conditions for testing. For the Testing Commands, the OGF is defined as 0x06.

11.7.6.1 Read Loopback Mode command

This command will read the value for the setting of the controller's loopback mode. The setting of the loopback mode will determine the path of information. In nontesting mode operation, the loopback mode is set to nontesting mode, and the path of the information is as specified by this standard. In local loopback mode, every data packet (ACL, SCO, and eSCO) and command packet that is sent from the host to the controller is sent back with no modifications by the controller, as shown in Figure 97 (in 11.7.6.2). See Table 396 and Table 397. For details of loopback modes, see 11.7.6.2.

Table 396—Read Loopback Mode command

Command	OCF	Command parameters	Return parameters
HCI_Read_Loopback_Mode	0x0001	None	Status, Loopback_Mode

Table 397—Read Loopback Mode command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Loopback Mode command succeeded.
		0x01–0xFF	Read Loopback Mode command failed. See 10.3 for list of error codes.
Loopback_ Mode	1 octet	0x00	No loopback mode enabled. (default)
		0x01	Enable local loopback.
		0x02	Enable remote loopback.
		0x03–0xFF	Reserved for future use.

When the Read Loopback Mode command has completed, a Command Complete event will be generated.

11.7.6.2 Write Loopback Mode command

This command will write the value for the setting of the controller's loopback mode. The setting of the loopback mode will determine the path of information. In nontesting mode operation, the loopback mode is set to nontesting mode and the path of the information as specified by this standard. In local loopback mode, every data packet (ACL, SCO, and eSCO) and command packet that is sent from the host to the controller is sent back with no modifications by the controller, as shown in Figure 97.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

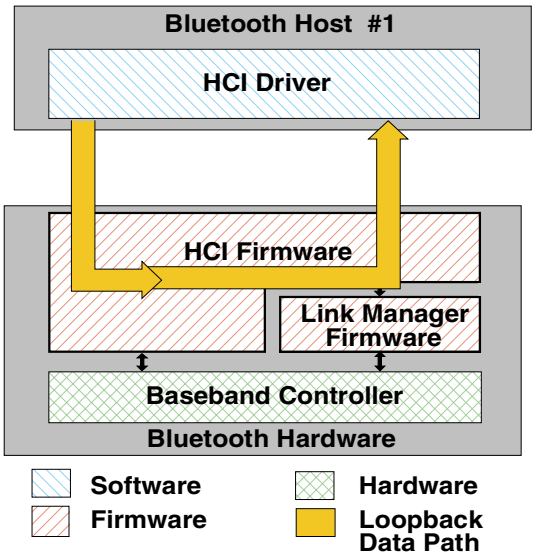


Figure 97—Local loopback mode

When the host controller enters local loopback mode, it shall respond with one to four connection handles, one for an ACL connection and zero to three for synchronous connections. The host should use these connection handles when sending data in local loopback mode. The number of connection handles returned for synchronous connections (between zero and three) is implementation specific. When in local loopback mode, the controller loops back commands and data to the host. The Loopback Command event is used to loop back commands that the host sends to the controller.

There are some commands that are not looped back in local loopback mode: Reset, Set Controller To Host Flow Control, Host Buffer Size, Host Number Of Completed Packets, Read Buffer Size, Read Loopback Mode, and Write Loopback Mode. These commands should be executed in the way they are normally executed. The Reset and Write Loopback Mode commands can be used to exit local loopback mode.

If the Write Loopback Mode command is used to exit local loopback mode, Disconnection Complete events corresponding to the Connection Complete events that were sent when entering local loopback mode should be sent to the host. Furthermore, no connections are allowed in local loopback mode. If there is a connection and there is an attempt to set the device to local loopback mode, the attempt will be refused. When the device is in local loopback mode, the controller will refuse incoming connection attempts. This allows the host controller transport layer to be tested without any other variables.

If a device is set to remote loopback mode, it will send back all data (ACL, SCO, and eSCO) that come over the air. It will allow only a maximum of one ACL connection and three synchronous connections, and these shall all be to the same remote device. If there are existing connections to a remote device and there is an attempt to set the local device to remote loopback mode, the attempt shall be refused.

See Figure 98, where the rightmost device is set to remote loopback mode and the leftmost device is set to nontesting mode. This allows the IEEE 802.15.1-2005 air link to be tested without any other variables. See also Table 398, Table 399, and Table 400.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

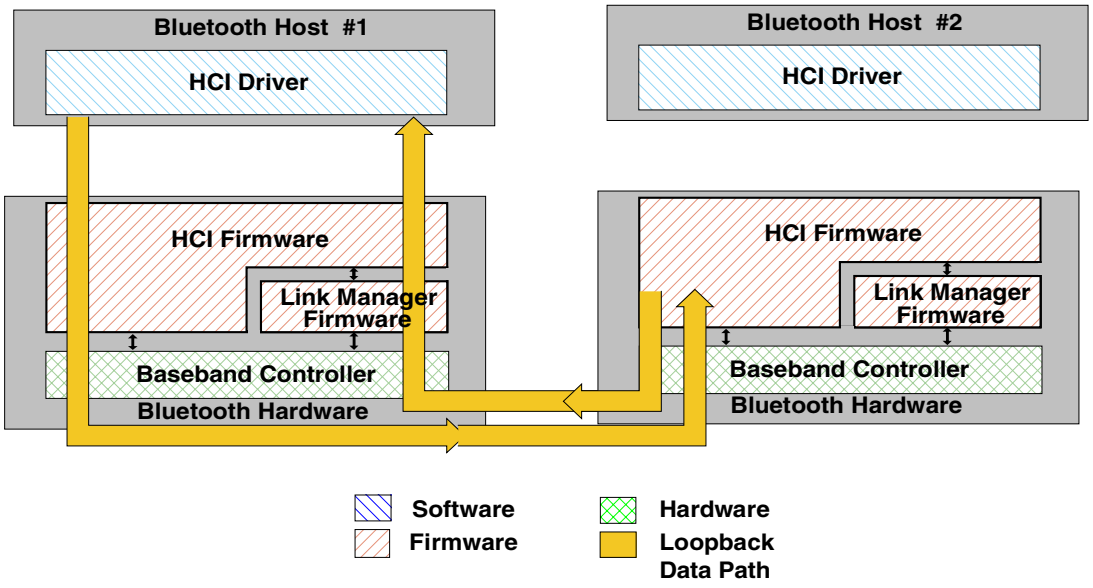


Figure 98—Remote loopback mode

Table 398—Write Loopback Mode command

Command	OCF	Command parameters	Return parameters
HCI_Write_Loopback_Mode	0x0002	Loopback_Mode	Status

Table 399—Write Loopback Mode command parameter

Parameter	Size	Value	Parameter description
Loopback_ Mode	1 octet	0x00	No loopback mode enabled. (default)
		0x01	Enable local loopback.
		0x02	Enable remote loopback.
		0x03–0xFF	Reserved for future use.

Table 400—Write Loopback Mode command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Loopback Mode command succeeded.
		0x01–0xFF	Write Loopback Mode command failed. See 10.3 for list of error codes.

When the Write Loopback Mode command has completed, a Command Complete event will be generated.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.6.3 Enable Device Under Test Mode command

The Enable Device Under Test Mode command will allow the local module to enter test mode via LMP test commands. For details, see Clause 9. When the controller receives this command, it will complete the command with a Command Complete event. The controller functions as normal until the remote tester issues the LMP test command to place the local device into DUT mode. To disable and exit the DUT mode, the host can issue the Reset command. This command prevents remote devices from causing the local device to enter test mode without first issuing this command. See Table 401 and Table 402.

Table 401—Enable Device Under Test Mode command

Command	OCF	Command parameters	Return parameters
HCI_Enable_Device_Under_Test_Mode	0x0003	None	Status

Table 402—Enable Device Under Test Mode command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Enter Device Under Test Mode command succeeded.
		0x01–0xFF	Enter Device Under Test Mode command failed. See 10.3 for list of error codes.

When the Enter Device Under Test Mode command has completed, a Command Complete event will be generated.

11.7.7 Events

11.7.7.1 Inquiry Complete event

The Inquiry Complete event indicates that the inquiry is finished. This event contains a status parameter, which is used to indicate if the inquiry completed successfully or if the inquiry was not completed. See Table 403 and Table 404.

Table 403—Inquiry Complete event

Event	Event code	Event parameters
Inquiry Complete	0x01	Status

Table 404—Inquiry Complete event parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Inquiry command completed successfully.
		0x01–0xFF	Inquiry command failed. See 10.3 for list of error codes.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.2 Inquiry Result event

The Inquiry Result event indicates that a device or multiple devices have responded so far during the current inquiry process. This event will be sent from the controller to the host as soon as an inquiry response from a remote device is received if the remote device supports only mandatory paging scheme. The controller may queue these inquiry responses and send multiple devices information in one Inquiry Result event. The event can be used to return one or more inquiry responses in one event. See Table 405 and Table 406.

Table 405—Inquiry Result event

Event	Event code	Event parameters
Inquiry Result	0x02	Num_Responses, BD_ADDR[i], Page_Scan_Repetition_Mode[i], Page_Scan_Period_Mode[i], Reserved[i], Class_of_Device[i] Clock_Offset[i]

Table 406—Inquiry result event parameters

Parameter	Size	Value	Parameter description
Num_ Responses	1 octet	0xXX	Number of responses from the inquiry.
BD_ADDR[i]	6 octets * Num_ Responses	0xXXXXXXXXXXX	BD_ADDR for each device which responded.
Page_Scan_ Repetition_ Mode[i]	1 octet * Num_ Responses	0x00	R0.
		0x01	R1.
		0x02	R2.
		0x03–0xFF	Reserved.
Page_Scan_ Period_Mode[i]	1 octet * Num_ Responses	0x00	P0.
		0x01	P1.
		0x02	P2.
		0x03–0xFF	Reserved.
Reserved[i]	1 octet * Num_ Responses	0xXX	Reserved; must be set to 0x00.
Class_of_ Device[i]	3 octets * Num_ Responses	0xXXXXXX	Class of the device.
Clock_Offset[i]	2 octets * Num_ Responses	Bit 14–0	Bit 16–2 of CLK_{slave}–CLK_{master}.
		Bit 15	Reserved.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.3 Connection Complete event

The Connection Complete event indicates to both of the hosts forming the connection that a new connection has been established. This event also indicates to the host that issued the Create Connection, Accept Connection Request, or Reject Connection Request command and then received a Command Status event, if the issued command failed or was successful. See Table 407 and Table 408.

Table 407—Connection Complete event

Event	Event code	Event parameters
Connection Complete	0x03	Status, Connection_Handle, BD_ADDR, Link_Type, Encryption_Mode

Table 408—Connection Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Connection successfully completed.
		0x01–0xFF	Connection failed to complete. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection between two devices. The connection handle is used as an identifier for transmitting and receiving voice or data. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
BD_ADDR	6 octets	0xXXXXXXXXXXX	BD_ADDR of the other connected device forming the connection.
Link_Type	1 octet	0x00	SCO connection.
		0x01	ACL connection (data channels).
		0x02–0xFF	Reserved for future use.
Encryption_ Mode	1 octet		See 11.6.16.

11.7.7.4 Connection Request event

The Connection Request event is used to indicate that a new incoming connection is trying to be established. The connection may be either accepted or rejected. If this event is masked away, if there is an incoming connection attempt, and if the controller is not set to automatically accept this connection attempt, the controller will automatically refuse the connection attempt. When the host receives this event and the Link_Type parameter is ACL, it should respond with either an Accept Connection Request or Reject Connection Request command before the Conn_Accept_Timeout timer expires. If the link type is SCO or eSCO, the host should reply with the Accept Synchronous Connection Request or the Reject Synchronous Connection Request command. If the link type is SCO, the host may respond with an Accept Connection Request command. If the event is responded to with an Accept Connection Request command, then the default parameter settings of the Accept Synchronous Connection Request command (see 11.7.1.27) should be used by the local LM when negotiating the SCO or eSCO link parameters. In that case, the Connection Complete event,

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

and not the Synchronous Connection Complete event, shall be returned on completion of the connection. See Table 409 and Table 410.

Table 409—Connection Request event

Event	Event code	Event parameters
Connection Request	0x04	BD_ADDR, Class_of_Device, Link_Type

Table 410—Connection request event parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXXXXXXXX	BD_ADDR of the device that requests the connection.
Class_of_Device	3 octets	0xXXXXXX	Class of the device that requests the connection.
		0x000000	Unknown class of device.
Link_Type	1 octet	0x00	SCO connection requested.
		0x01	ACL connection requested.
		0x02	eSCO connection requested.
		0x03–0xFF	Reserved for future use.

11.7.7.5 Disconnection Complete event

The Disconnection Complete event occurs when a connection is terminated. The status parameter indicates if the disconnection was successful or not. The Reason parameter indicates the reason for the disconnection if the disconnection was successful. If the disconnection was not successful, the value of the Reason parameter can be ignored by the host. For example, this can be the case if the host has issued the Disconnect command when there was a parameter error, the command was not presently allowed, or a connection handle that did not correspond to a connection was given. See Table 411 and Table 412.

Table 411—Disconnection Complete event

Event	Event code	Event parameters
Disconnection Complete	0x05	Status, Connection_Handle, Reason

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 412—Disconnection Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Disconnection has occurred.
		0x01–0xFF	Disconnection failed to complete. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle that was disconnected. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Reason	1 octet	0xXX	Reason for disconnection. See 10.3 for error codes and description.

When a physical link fails, one Disconnection Complete event will be returned for each logical channel on the physical link with the corresponding connection handle as a parameter.

11.7.7.6 Authentication Complete event

The Authentication Complete event occurs when authentication has been completed for the specified connection. The connection handle must be a connection handle for an ACL connection. See Table 413 and Table 414.

Table 413—Authentication Complete event

Event	Event code	Event parameters
Authentication Complete	0x06	Status, Connection_Handle

Table 414—Authentication Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Authentication request successfully completed.
		0x01–0xFF	Authentication request failed to complete. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle for which authentication has been performed. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

11.7.7.7 Remote Name Request Complete event

The Remote Name Request Complete event is used to indicate that a remote name request has been completed. The Remote_Name event parameter is a UTF-8 encoded string with up to 248 octets in length. The Remote_Name event parameter will be null-terminated (0x00) if the UTF-8 encoded string is less than 248 octets. The BD_ADDR event parameter is used to identify from which device the user-friendly name was obtained. See Table 415 and Table 416.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 415—Remote Name Request Complete event

Event	Event code	Event parameters
Remote Name Request Complete	0x07	Status, BD_ADDR, Remote_Name

Table 416—Remote Name Request Complete Event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Remote Name Request command succeeded.
		0x01–0xFF	Remote Name Request command failed. See 10.3 for list of error codes.
BD_ADDR	6 octets	0xXXXXXXXXXXX	BD_ADDR for the device whose name was requested.
Remote_ Name	248 octets	Name[248]	A UTF-8 encoded user-friendly descriptive name for the remote device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

The Remote_Name parameter is a string parameter. Endianess does not, therefore, apply to the Remote_Name parameter. The first octet of the name is received first.

11.7.7.8 Encryption Change event

The Encryption Change event is used to indicate that the change in the encryption has been completed for the connection handle specified by the Connection_Handle event parameter. The connection handle will be a connection handle for an ACL connection. The Encryption_Enable event parameter specifies the new encryption for the connection handle specified by the Connection_Handle event parameter. This event will occur on both devices to notify the hosts when encryption has changed for the specified connection handle between two devices. See Table 417 and Table 418.

Table 417—Encryption Change event

Event	Event code	Event parameters
Encryption Change	0x08	Status, Connection_Handle, Encryption_Enable

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 418—Encryption change event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Encryption change has occurred.
		0x01–0xFF	Encryption change failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle for which the link-layer encryption has been enabled/disabled for all connection handles with the same device endpoint as the specified connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Encryption_ Enable	1 octet	0x00	Link-level encryption is off.
		0x01	Link-level encryption is on.

11.7.7.9 Change Connection Link Key Complete event

The Change Connection Link Key Complete event is used to indicate that the change in the link key for the connection handle specified by the Connection_Handle event parameter has been completed. The connection handle will be a connection handle for an ACL connection. The Change Connection Link Key Complete event is sent only to the host that issued the Change Connection Link Key command. See Table 419 and Table 420.

Table 419—Change Connection Link Key Complete event

Event	Event code	Event parameters
Change Connection Link Key Complete	0x09	Status, Connection_Handle

Table 420—Change Connection Link Key Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Change Connection Link Key command succeeded.
		0x01–0xFF	Change Connection Link Key command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle that the link key has been change for all connection handles with the same device endpoint as the specified connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

11.7.7.10 Master Link Key Complete event

The Master Link Key Complete event is used to indicate that the link key managed by the master of the piconet has been changed. The connection handle will be a connection handle for an ACL connection. The link key used for the connection will be the temporary link key of the master device or the semi-permanent link key indicated by the Key_Flag event parameter. The Key_Flag event parameter is used to indicate which

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

link key (the temporary link key of the master or the semi-permanent link keys) is now being used in the piconet. See Table 421 and Table 422.

Table 421—Master Link Key Complete event

Event	Event code	Event parameters
Master Link Key Complete	0x0A	Status, Connection_Handle, Key_Flag

Table 422—Master Link Key Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Master Link Key command succeeded.
		0x01–0xFF	Master Link Key command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle for which the link key has been changed for all devices in the same piconet. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Key_Flag	1 octet	0x00	Using semi-permanent link key.
		0x01	Using temporary link key.

For a master, the change from a semi-permanent link key to temporary link key will affect all connection handles related to the piconet. For a slave, this change affects only this particular connection handle. A temporary link key must be used when both broadcast and point-to-point traffic shall be encrypted.

11.7.7.11 Read Remote Supported Features Complete event

The Read Remote Supported Features Complete event is used to indicate the completion of the process of when the LM obtains the supported features of the remote device specified by the Connection_Handle event parameter. The connection handle will be a connection handle for an ACL connection. The event parameters include a list of LMP features. See Table 423 and Table 424. For details, see Clause 9.

Table 423—Read Remote Supported Features Complete event

Event	Event code	Event parameters
Read Remote Supported Features Complete	0x0B	Status, Connection_Handle, LMP_Features

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 424—Read Remote Supported Features Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Remote Supported Features command succeeded.
		0x01–0xFF	Read Remote Supported Features command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle that is used for the Read Remote Supported Features command. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
LMP_ Features	8 octets	0xXXXXX XXXXXX XXXXX	Bit mask list of LMP features. See Clause 9.

11.7.7.12 Read Remote Version Information Complete event

The Read Remote Version Information Complete event is used to indicate the completion of the process of when the LM obtains the version information of the remote device specified by the Connection_Handle event parameter. The connection handle will be a connection handle for an ACL connection. The LMP_Version event parameter defines the specification version of the device. The Manufacturer_Name event parameter indicates the manufacturer of the remote device. The LMP_Subversion event parameter is controlled by the manufacturer and is implementation dependent. The LMP_Subversion event parameter defines the various revisions that each version of the hardware will go through as design processes change and errors are fixed. This allows the software to determine what hardware is being used and, if necessary, to work around various bugs in the hardware. See Table 425 and Table 426.

Table 425—Read Remote Version Information Complete event

Event	Event code	Event parameters
Read Remote Version Information Complete	0x0C	Status, Connection_Handle, LMP_Version, Manufacturer_Name, LMP_Subversion

Table 426—Read Remote Version Information Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Remote Version Information command succeeded.
		0x01–0xFF	Read Remote Version Information command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle that is used for the Read Remote Version Information command. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
LMP_Version	1 octet	0xXX	Version of the current LMP in the remote device. For LMP version information, see Bluetooth Assigned Numbers [B1].

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 426—Read Remote Version Information Complete event parameters *(continued)*

Parameter	Size	Value	Parameter description
Manufacturer_ Name	2 octets	0xXXXX	Manufacturer name of the remote device. See Table 68 for assigned values (CompId).
LMP_ Subversion	2 octets	0xXXXX	Subversion of the current LMP in the remote device. See Table 68 for assigned values (SubVersNr).

11.7.7.13 QoS Setup Complete event

The QoS Setup Complete event is used to indicate the completion of the process of when the LM sets up QoS with the remote device specified by the Connection_Handle event parameter. The connection_ handle will be a connection handle for an ACL connection. For more detail, see Clause 14. See Table 427 and Table 428.

Table 427—QoS Setup Complete event

Event	Event code	Event parameters
QoS Setup Complete	0x0D	Status, Connection_Handle, Flags, Service_Type, Token_Rate, Peak_Bandwidth, Latency, Delay_Variation

Table 428—QoS Setup Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	QoS Setup command succeeded.
		0x01–0xFF	QoS Setup command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle that is used for the QoS Setup command. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Flags	1 octet	0x00–0xFF	Reserved for future use.
Service_ Type	1 octet	0x00	No traffic available.
		0x01	Best effort available.
		0x02	Guaranteed available.
		0x03–0xFF	Reserved for future use.
Token_ Rate	4 octets	0xXXXXXXXX	Available token rate, in octets per second.
Peak_ Bandwidth	4 octets	0xXXXXXXXX	Available peak bandwidth, in octets per second.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 428—QoS Setup Complete event parameters *(continued)*

Parameter	Size	Value	Parameter description
Latency	4 octets	0xXXXXXXXX	Available latency, in microseconds.
Delay_ Variation	4 octets	0xXXXXXXXX	Available delay variation, in microseconds.

11.7.7.14 Command Complete event

The Command Complete event is used by the controller for most commands to transmit return status of a command and the other event parameters that are specified for the issued HCI command.

The Num_HCI_Command_Packets event parameter allows the controller to indicate the number of HCI command packets the host can send to the controller. If the controller requires the host to stop sending commands, the Num_HCI_Command_Packets event parameter will be set to zero. To indicate to the host that the controller is ready to receive HCI command packets, the controller generates a Command Complete event with the command opcode 0x0000 and with the Num_HCI_Command_Packets event parameter set to 1 or more. Command opcode 0x0000 is a NOP and can be used to change the number of outstanding HCI command packets that the host can send before waiting. See each command for the parameters that are returned by this event. See Table 429 and Table 430.

Table 429—Command Complete event

Event	Event code	Event parameters
Command Complete	0x0E	Num_HCI_Command_Packets, Command_Opcode, Return_Parameters

Table 430—Command Complete event parameters

Parameter	Size	Value	Parameter description
Num_HCI_ Command_ Packets	1 octet	*N* = 0xXX	The number of HCI command packets which are allowed to be sent to the controller from the host. Range for *N*: 0–255
Command_ Opcode	2 octets	0xXXXX	Opcode of the command that caused this event.
Return_ Parameters	Depends on command	0xXX	This is the return parameter(s) for the command specified in the Command_Opcode event parameter. See each command's definition for the list of return parameters associated with that command.

11.7.7.15 Command Status event

The Command Status event is used to indicate that the command described by the Command_Opcode parameter has been received and that the controller is currently performing the task for this command. This

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

event is needed to provide mechanisms for asynchronous operation, which makes it possible to prevent the host from waiting for a command to finish. If the command cannot begin to execute (e.g., a parameter error may have occurred or the command may currently not be allowed), the Status event parameter will contain the corresponding error code, and no complete event will follow since the command was not started. The Num_HCI_Command_Packets event parameter allows the controller to indicate the number of HCI command packets the host can send to the controller. If the controller requires the host to stop sending commands, the Num_HCI_Command_Packets event parameter will be set to zero. To indicate to the host that the controller is ready to receive HCI command packets, the controller generates a Command Status event with status 0x00 and command opcode 0x0000 and with the Num_HCI_Command_Packets event parameter set to 1 or more. Command opcode 0x0000 is a NOP and can be used to change the number of outstanding HCI command packets that the host can send before waiting. See Table 431 and Table 432.

Table 431—Command Status event

Event	Event code	Event parameters
Command Status	0x0F	Status, Num_HCI_Command_Packets, Command_Opcode

Table 432—Command Status Event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Command currently is pending.
		0x01–0xFF	Command failed. See 10.3 for list of error codes.
Num_HCI_Command_Packets	1 octet	*N* = 0xXX	The number of HCI command packets that are allowed to be sent to the controller from the host. Range for *N*: 0–255
Command_Opcode	2 octets	0xXXXX	Opcode of the command that caused this event and is pending completion.

11.7.7.16 Hardware Error event

The Hardware Error event is used to indicate some type of hardware failure for the device. This event is used to notify the host that a hardware failure has occurred in the device. See Table 433 and Table 434.

Table 433—Hardware Error event

Event	Event code	Event parameters
Hardware Error	0x10	Hardware_Code

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 434—Hardware Error event parameter

Parameter	Size	Value	Parameter description
Hardware_ Code	1 octet	0x00–0xFF	These hardware codes will be implementation specific and can be assigned to indicate various hardware problems.

11.7.7.17 Flush Occurred event

The Flush Occurred event is used to indicate that, for the specified connection handle, the current user data to be transmitted have been removed. The connection handle will be a connection handle for an ACL connection. This could result from the flush command or be due to the automatic flush. Multiple blocks of an L2CAP packet could have been pending in the controller. If one BB packet part of an L2CAP packet is flushed, then the rest of the HCI data packets for the L2CAP packet must also be flushed. See Table 435 and Table 436.

Table 435—Flush Occurred event

Event	Event code	Event parameters
Flush Occurred	0x11	Connection_Handle

Table 436—Flush Occurred event parameter

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle that was flushed. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

11.7.7.18 Role Change event

The Role Change event is used to indicate that the current role related to the particular connection has changed. This event occurs only when both the remote and local devices have completed their role change for the device associated with the BD_ADDR event parameter. This event allows both affected hosts to be notified when the role has been changed. See Table 437 and Table 438.

Table 437—Role Change event

Event	Event code	Event parameters
Role Change	0x12	Status, BD_ADDR, New_Role

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 438—Role Change event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Role change has occurred.
		0x01–0xFF	Role change failed. See 10.3 for list of error codes.
BD_ADDR	6 octets	0xXXXXXXXXXXX	BD_ADDR of the device for which a role change has completed.
New_Role	1 octet	0x00	Currently the master for specified BD_ADDR.
		0x01	Currently the slave for specified BD_ADDR.

11.7.7.19 Number Of Completed Packets event

The Number Of Completed Packets event is used by the controller to indicate to the host how many HCI data packets have been completed (transmitted or flushed) for each connection handle since the previous Number Of Completed Packets event was sent to the host. This means that the corresponding buffer space has been freed in the controller. Based on this information and on the HC_Total_Num_ACL_Data_Packets and HC_Total_Num_Synchronous_Data_Packets return parameter of the Read Buffer Size command, the host can determine for which connection handles the following HCI data packets should be sent to the controller. The Number Of Completed Packets event must not be sent before the corresponding Connection Complete event. While the controller has HCI data packets in its buffer, it must keep sending the Number Of Completed Packets event to the host at least periodically until it finally reports that all the pending ACL data packets have been transmitted or flushed. The rate with which this event is sent is manufacturer specific. See Table 439 and Table 440.

Table 439—Number Of Completed Packets event

Event	Event code	Event parameters
Number Of Completed Packets	0x13	Number_of_Handles, Connection_Handle[i], HC_Num_Of_Completed_Packets[i]

Table 440—Number Of Completed Packets event parameters

Parameter	Size	Value	Parameter description
Number_of_ Handles	1 octet	0xXX	The number of connection handle and HCI data packet pairs contained in this event. Range: 0–255
Connection_ Handle[i]	Number_of_ Handles * 2 octets	0xXXXX	Connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
HC_Num_Of_ Completed_ Packets [i]	Number_of_ Handles * 2 octets	*N* = 0xXXXX	The number of HCI data packets that have been completed (transmitted or flushed) for the associated connection handle since the previous time the event was returned. Range for *N*: 0x0000–0xFFFF

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The Number Of Completed Packets events will not report on synchronous connection handles if synchronous flow control is disabled. (For details on the Read/Write Synchronous Flow Control Enable commands, see 11.7.3.38 and 11.7.3.39.)

11.7.7.20 Mode Change event

The Mode Change event is used to indicate when the device associated with the connection handle changes between active mode, HOLD mode, SNIFF mode, and PARK state. The connection handle will be a connection handle for an ACL connection. The Connection_Handle event parameter is used to indicate with which connection the Mode Change event is associated. The Current_Mode event parameter is used to indicate the state in which the connection is currently. The Interval parameter is used to specify a time amount specific to each state. Each controller that is associated with the connection handle that has changed modes will send the Mode Change event to its host. See Table 441 and Table 442.

Table 441—Mode Change event

Event	Event code	Event parameters
Mode Change	0x14	Status, Connection_Handle, Current_Mode, Interval

Table 442—Mode Change event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	A mode change has occurred.
		0x01–0xFF	Hold Mode, Sniff Mode, Exit Sniff Mode, Park State, or Exit Park State command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Current_ Mode	1 octet	0x00	Active mode.
		0x01	HOLD mode.
		0x02	SNIFF mode.
		0x03	PARK state.
		0x04-0xFF	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 442—Mode Change event parameters *(continued)*

Parameter	Size	Value	Parameter description
Interval	2 octets	*N* = 0xXXXX	HOLD: Number of BB slots to wait in HOLD mode. Hold interval = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0002–0xFFFE Time range: 1.25 ms to 40.9 s SNIFF: Number of BB slots between sniff intervals. Time between sniff intervals = 0.625 ms (1 BB slot) Range for *N*: 0x0002–0xFFFE Time range: 1.25 ms to 40.9 s PARK: Number of BB slots between consecutive beacons. Interval length = *N* * 0.625 ms (1 BB slot) Range for *N*: 0x0002–0xFFFE Time range: 1.25 ms to 40.9 s

11.7.7.21 Return Link Keys event

The Return Link Keys event is used by the controller to send the host one or more stored link keys. Zero or more instances of this event will occur after the Read Stored Link Key command. When there are no link keys stored, no Return Link Keys events will be returned. When there are link keys stored, the number of link keys returned in each Return Link Keys event is implementation specific. See Table 443 and Table 444.

Table 443—Return Link Keys event

Event	Event code	Event parameters
Return Link Keys	0x15	Num_Keys, BD_ADDR [i], Link_Key[i]

Table 444—Return Link Keys event parameters

Parameter	Size	Value	Parameter description
Num_Keys	1 octet	0xXX	Number of link keys contained in this event. Range: 0x01–0x0B
BD_ADDR [i]	6 octets * Num_Keys	0xXXXXXXXXXXX	BD_ADDR for the associated link key.
Link_Key[i]	16 octets * Num_Keys	0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Link key for the associated BD_ADDR.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.22 PIN Code Request event

The PIN Code Request event is used to indicate that a PIN code is required to create a new link key. The host must respond using either the PIN Code Request Reply or the PIN Code Request Negative Reply command, depending on whether the host can provide the controller with a PIN code.

If the PIN Code Request event is masked away, then the controller will assume that the host has no PIN code.

When the controller generates a PIN Code Request event in order for the local LM to respond to the request from the remote LM (as a result of a Create Connection or Authentication Requested command from the remote host), the local host must respond with either a PIN Code Request Reply or PIN Code Request Negative Reply command before the remote LM detects LMP response timeout. (See Clause 9.) See Table 445 and Table 446.

Table 445—Pin Code Request event

Event	Event code	Event parameters
PIN Code Request	0x16	BD_ADDR

Table 446—Pin Code Request event parameter

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which a new link key is being created.

11.7.7.23 Link Key Request event

The Link Key Request event is used to indicate that a link key is required for the connection with the device specified in BD_ADDR. If the host has the requested stored link key, then the host will pass the requested key to the controller using the Link Key Request Reply command. If the host does not have the requested stored link key, then the host will use the Link Key Request Negative Reply command to indicate to the controller that the host does not have the requested key.

If the Link Key Request event is masked away, then the controller will assume that the host has no additional link keys.

When the controller generates a Link Key Request event in order for the local LM to respond to the request from the remote LM (as a result of a Create Connection or Authentication Requested command from the remote host), the local host must respond with either a Link Key Request Reply or Link Key Request Negative Reply command before the remote LM detects LMP response timeout. (See Clause 9.) See Table 447 and Table 448.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 447—Link Key Request event

Event	Event code	Event parameters
Link Key Request	0x17	BD_ADDR

Table 448—Link Key Request event parameter

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which a stored link key is being requested.

11.7.7.24 Link Key Notification event

The Link Key Notification event is used to indicate to the host that a new link key has been created for the connection with the device specified in BD_ADDR. The host can save this new link key in its own storage for future use. Also, the host can decided to store the link key in the controller's link key storage by using the Write Stored Link Key command. The Key_Type event parameter informs the host about which key type (combination key, local unit key, or remote unit key) that has been used during pairing. If pairing with unit key is not supported, the host can, for instance, discard the key or disconnect the link. See Table 449 and Table 450.

Table 449—Link Key Notification event

Event	Event code	Event parameters
Link Key Notification	0x18	BD_ADDR, Link_Key, Key_Type

Table 450—Link Key Notification event parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the device for which the new link key has been generated.
Link_Key	16 octets	0xXXXXXX XXXXXXX XXXXXXX XXXXXXX XXXXX	Link key for the associated BD_ADDR.
Key_Type	1 octet	0x00	Combination key.
		0x01	Local unit key.
		0x02	Remote unit key.
		0x03–0xFF	Reserved.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.25 Loopback Command event

When in local loopback mode, the controller loops back commands and data to the host. The Loopback Command event is used to loop back all commands that the host sends to the controller with some exceptions. See 11.7.6.1 for a description of which commands that are not looped back. The HCI_Command_Packet event parameter contains the entire HCI command packet including the header. See Table 451 and Table 452.

Table 451—Loopback Command event

Event	Event code	Event parameters
Loopback Command	0x19	HCI_Command_Packet

Table 452—Loopback Command event parameters

Parameter	Size	Value	Parameter description
HCI_Command_ Packet	Depends on command	0xXXXXXX	HCI command packet, including header.

The event packet is limited to a maximum of 255 octets in the payload; since an HCI command packet has 3 octets of header data, only the first 252 octets of the command parameters will be returned.

11.7.7.26 Data Buffer Overflow event

This event is used to indicate that the controller's data buffers have been overflowed. This can occur if the host has sent more packets than allowed. The Link_Type parameter is used to indicate that the overflow was caused by ACL or synchronous data. See Table 453 and Table 454.

Table 453—Data Buffer Overflow event

Event	Event code	Event parameters
Data Buffer Overflow	0x1A	Link_Type

Table 454—Data Buffer Overflow event parameter

Parameter	Size	Value	Parameter description
Link_Type	1 octet	0x00	Synchronous buffer overflow (voice channels).
		0x01	ACL buffer overflow (data channels).
		0x02–0xFF	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.27 Max Slots Change event

This event is used to notify the host about the LMP_Max_Slots parameter when the value of this parameter changes. It will be sent each time the maximum allowed length, in number of slots, for BB packets transmitted by the local device changes. The connection handle will be a connection handle for an ACL connection. See Table 455 and Table 456.

Table 455—Max Slots Change event

Event	Event code	Event parameters
Max Slots Change	0x1B	Connection_Handle, LMP_Max_Slots

Table 456—Max Slots Change event parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
LMP_Max_ Slots	1 octet	0x01, 0x03, 0x05	Maximum number of slots allowed to use for BB packets (see 9.3.1.10 and 9.4.2).

11.7.7.28 Read Clock Offset Complete event

The Read Clock Offset Complete event is used to indicate the completion of the process of the LM obtaining the Clock Offset information of the device specified by the Connection_Handle event parameter. The connection handle will be a connection handle for an ACL connection. See Table 457 and Table 458.

Table 457—Read Clock Offset Complete event

Event	Event code	Event parameters
Read Clock Offset Complete	0x1C	Status, Connection_Handle, Clock_Offset

Table 458—Read Clock Offset Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Clock Offset command succeeded.
		0x01–0xFF	Read Clock Offset command failed. See 10.3 for list of error codes.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 458—Read Clock Offset Complete event parameters *(continued)*

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Specifies which connection handle's Clock_Offset parameter is returned. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Clock_ Offset	2 octets	Bit 14–0	Bit 16–2 of CLK_{slave}–CLK_{master}.
		Bit 15	Reserved.

11.7.7.29 Connection Packet Type Changed event

The Connection Packet Type Changed event is used to indicate the completion of the process of when the LM changes which packet types can be used for the connection. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type event parameter specifies which packet types the LM can use for the connection identified by the Connection_Handle event parameter for sending L2CAP data or voice. The Packet_Type event parameter does not decide which packet types the LM is allowed to use for sending LMP PDUs. See Table 459 and Table 460.

Table 459—Connection Packet Type Changed event

Event	Event code	Event parameters
Connection Packet Type Changed	0x1D	Status, Connection_Handle, Packet_Type

Table 460—Connection Packet Type Changed event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Connection packet type changed successfully.
		0x01–0xFF	Connection packet type change failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Packet_Type for ACL link type	2 octets	0x0001	Reserved for future use.
		0x0002	Reserved for future use.
		0x0004	Reserved for future use.
		0x0008	**DM1**
		0x0010	**DH1**
		0x0020	Reserved for future use.
		0x0040	Reserved for future use.
		0x0080	Reserved for future use.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 460—Connection Packet Type Changed event parameters *(continued)*

Parameter	Size	Value	Parameter description
Packet_Type for ACL link type *(continued)*		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	**DM3**
		0x0800	**DH3**
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	**DM5**
		0x8000	**DH5**
Packet_Type for SCO link type		0x0001	Reserved for future use.
		0x0002	Reserved for future use.
		0x0004	Reserved for future use.
		0x0008	Reserved for future use.
		0x0010	Reserved for future use.
		0x0020	**HV1**
		0x0040	**HV2**
		0x0080	**HV3**
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	Reserved for future use.
		0x0800	Reserved for future use.
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	Reserved for future use.
		0x8000	Reserved for future use.

11.7.7.30 QoS Violation event

The QoS Violation event is used to indicate that the LM is unable to provide the current QoS requirement for the connection handle. This event indicates that the LM is unable to provide one or more of the agreed QoS parameters. The host chooses what action should be done. The host can reissue QoS Setup command to renegotiate the QoS setting for connection handle. The connection handle will be a connection handle for an ACL connection. See Table 461 and Table 462.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 461—QoS Violation event

Event	Event code	Event parameters
QoS Violation	0x1E	Connection_Handle

Table 462—QoS Violation event parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle for which the LM is unable to provide the current QoS requested. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)

11.7.7.31 Page Scan Repetition Mode Change event

The Page Scan Repetition Mode Change event indicates that the remote device with the specified BD_ADDR has successfully changed the page scan repetition (SR) mode. See Table 463 and Table 464.

Table 463—Page Scan Repetition Mode Change event

Event	Event code	Event parameters
Page Scan Repetition Mode Change	0x20	BD_ADDR, Page_Scan_Repetition_Mode

Table 464—Page Scan Repetition Mode Change event parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the remote device.
Page_Scan_ Repetition_ Mode	1 octet	0x00	R0.
		0x01	R1.
		0x02	R2.
		0x03–0xFF	Reserved.

11.7.7.32 Flow Specification Complete event

The Flow Specification Complete event is used to inform the host about the QoS for the ACL connection the controller is able to support. The connection handle will be a connection handle for an ACL connection. The flow parameters refer to the outgoing or incoming traffic of the ACL link, as indicated by the Flow_Direction parameter. The flow parameters are defined in 14.5.3. When the Status parameter indicates a successful completion, the flow parameters specify the agreed values by the controller. When the Status parameter indicates a failed completion with the error code *QoS unacceptable parameters* (0x2C), the flow parameters specify the acceptable values of the controller. This enables the host to continue the QoS negotiation with a new HCI Flow Specification command with flow parameter values that are acceptable for the

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

controller. When the Status parameter indicates a failed completion with the error code *QoS rejected* (0x2D), this indicates a request of the controller to discontinue the QoS negotiation. When the Status parameter indicates a failed completion, the flow parameter values of the most recently successful completion must be assumed (or the default values when there was no successful completion). See Table 465 and Table 466.

Table 465—Flow specification complete event

Event	Event code	Event parameters
HCI Flow Specification Complete	0x21	Status, Connection_Handle, Flags, Flow_Direction, Service_Type, Token_Rate, Token_Bucket_Size, Peak_Bandwidth, Access_Latency

Table 466—Flow Specification Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Flow Specification command succeeded.
		0x01–0xFF	Flow Specification command failed. See 10.3 for list of error codes.
Connection_ Handle	2 octets	0xXXXX	Connection handle used to identify for which ACL connection the flow is specified. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Flags	1 octet	0x00–0xFF	Reserved for future use.
Flow_Direction	1 octet	0x00	Outgoing flow, i.e., traffic send over the ACL connection.
		0x01	Incoming flow, i.e., traffic received over the ACL connection.
		0x02–0xFF	Reserved for future use.
Service_Type	1 octet	0x00	No traffic.
		0x01	Best effort.
		0x02	Guaranteed.
		0x03–0xFF	Reserved for future use.
Token_Rate	4 octets	0xXXXXX XXX	Token rate, in octets per second.
Token_Bucket_ Size	4 octets	0xXXXXX XXX	Token bucket size, in octets.
Peak_ Bandwidth	4 octets	0xXXXXX XXX	Peak bandwidth, in octets per second.
Access_Latency	4 octets	0xXXXXX XXX	Access latency, in microseconds.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.33 Inquiry Result With RSSI event

The Inquiry Result With RSSI event indicates that a device or multiple devices have responded so far during the current inquiry process. This event will be sent from the controller to the host as soon as an inquiry response from a remote device is received if the remote device supports only mandatory paging scheme. This controller may queue these inquiry responses and send multiple devices information in one Inquiry Result event. The event can be used to return one or more Inquiry responses in one event. The RSSI[i] parameter is measured during the FHS packet returned by each responding slave. See Table 467 and Table 468.

Table 467—Inquiry Result With RSSI event

Event	Event code	Event parameters
Inquiry Result with RSSI	0x22	Num_responses, BD_ADDR[i], Page_Scan_Repetition_Mode[i], Page_Scan_Period_Mode[i], Class_of_Device[i], Clock_Offset[i], RSSI[i]

Table 468—Inquiry Result With RSSI event parameters

Parameter	Size	Value	Parameter description
Num_ Responses	1 octet	0xXX	Number of responses from the inquiry.
BD_ADDR[i]	6 octets * Num_ Responses	0xXXXXXX XXXXXX	BD_ADDR for each device which responded.
Page_Scan_ Repetition_Mode[i]	1 octet* Num_ Responses	0x00	R0.
		0x01	R1.
		0x02	R2.
		0x03–0xFF	Reserved.
Page_Scan_ Period_Mode[i]	1 octet* Num_ Responses	0x00	P0.
		0x01	P1.
		0x02	P2.
		0x03–0xFF	Reserved.
Class_of_Device[i]	3 octets * Num_ Responses	0xXXXXXX	Class of the device
Clock_Offset[i]	2 octets * Num_ Responses	Bit 14-0	Bit 16–2 of CLK_{slave}–CLK_{master}.
		Bit 15	Reserved.
RSSI[i]	1 octet * Num_ Responses	0xXX	Range: –127 to +20 Units: dBm

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

This event shall be generated only if the Inquiry_Mode parameter of the last Write Inquiry Mode command was set to 0x01 (Inquiry Result With RSSI event format).

The only difference between the Inquiry Result With RSSI event and the Inquiry Result event is the additional RSSI parameter.

11.7.7.34 Read Remote Extended Features Complete event

The Read Remote Extended Features Complete event is used to indicate the completion of the process of when the LM obtains the remote extended LMP features of the remote device specified by the connection handle event parameter. The connection handle will be a connection handle for an ACL connection. The event parameters include a page of the remote devices extended LMP features. See Table 469 and Table 470. For details, see Clause 9.

Table 469—Read Remote Extended Features Complete event

Event	Event code	Event parameters
Read Remote Extended Features Complete	0x23	Status, Connection_Handle, Page_Number, Maximum_Page_Number, Extended_LMP_Features

Table 470—Read Remote Extended Features Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Request for remote extended features succeeded
		0x01–0xFF	Request for remote extended features failed; standard HCI error value.
Connection_ Handle	2 octets	0xXXXX	The connection handle identifying the device to which the remote features apply. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Page_Number	1octet	0x00	The normal LMP features as returned by Read Remote Supported Features command.
		0x01–0xFF	The page number of the features returned.
Maximum_ Page_Number	1 octet	0x00–0xFF	The highest features page number which contains nonzero bits for the local device.
Extended_ LMP_Features	8 octets	0xFFFFFFFFFFFFFFFF	Bit map of requested page of LMP features. See Clause 9 for details.

11.7.7.35 Synchronous Connection Complete event

The Synchronous Connection Complete event indicates to both the hosts that a new synchronous connection has been established. This event also indicates to the host that issued the Setup Synchronous Connection, Accept Synchronous Connection Request, or Reject Synchronous Connection Request command and then

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

received a Command Status event if the issued command failed or was successful. See Table 471 and Table 472.

Table 471—Synchronous Connection Complete event

Event	Event code	Event parameters
Synchronous Connection Complete	0x2C	Status, Connection_Handle, BD_ADDR, Link_Type, Transmission_Interval, Retransmission_Window, Rx_Packet_Length, Tx_Packet_Length Air_Mode

Table 472—Synchronous Connection Complete event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Connection successfully completed.
		0x01–0xFF	Connection failed to complete. See 10.3 for error codes and description.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection between two devices. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the other connected device forming the connection.
Link_Type	1 octet	0x00	SCO connection.
		0x01	Reserved.
		0x02	eSCO connection.
		0x03–0xFF	Reserved.
Transmission_ Interval	1 octet	0xXX	Time between two consecutive eSCO instants measured in slots. Must be zero for SCO links.
Retransmission_ Window	1 octet	0xXX	The size of the retransmission window measured in slots. Must be zero for SCO links.
Rx_Packet_ Length	2 octets	0xXXXX	Length, in bytes, of the eSCO payload in the receive direction. Must be zero for SCO links.
Tx_Packet_ Length	2 octets	0xXXXX	Length, in bytes, of the eSCO payload in the transmit direction. Must be zero for SCO links.
Air_Mode	1 octet	0x00	μ-law log.
		0x01	A-law log.
		0x02	CVSD.
		0x03	Transparent data.
		0x04–0xFF	Reserved.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.7.7.36 Synchronous connection changed event

The Synchronous Connection Changed event indicates to the host that an existing synchronous connection has been reconfigured. This event also indicates to the initiating host (if the change was host initiated) if the issued command failed or was successful. See Table 473 and Table 474.

Table 473—Synchronous Connection Changed event

Event	Event code	Event parameters
Synchronous Connection Changed	0x2D	Status, Connection_Handle, Transmission_Interval, Retransmission_Window, Rx_Packet_Length, Tx_Packet_Length

Table 474—Synchronous Connection Changed event parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Connection successfully reconfigured.
		0x01–0xFF	Reconfiguration failed to complete. See 10.3 for error codes and description.
Connection_ Handle	2 octets	0xXXXX	Connection handle to be used to identify a connection between two devices. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Transmission_ Interval	1 octet	0xXX	Time between two consecutive SCO/eSCO instants measured in slots.
Retransmission_ Window	1 octet	0xXX	The size of the retransmission window measured in slots. Must be zero for SCO links.
Rx_Packet_ Length	2 octets	0xXXXX	Length, in bytes, of the SCO/eSCO payload in the receive direction.
Tx_Packet_ Length	2 octets	0xXXXX	Length, in bytes, of the SCO/eSCO payload in the transmit direction.

11.8 Deprecated commands, events, and configuration parameters

Commands, events, and configuration parameters in this subclause were in prior versions of this standard, but have been determined to be not required.

They may be implemented by a controller to allow for backwards compatibility with a host utilizing a prior version of the specification.

A host should not use these commands.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

11.8.1 Page_Scan_Mode parameter

The Page_Scan_Mode parameter indicates the page scan mode that is used for default page scan. Currently one mandatory page scan mode and three optional page scan modes are defined. Following an inquiry response, if the BB timer *T_mandatory_pscan* has not expired, the mandatory page scan mode must be applied. See Table 475. For details on page scan mode, FHS packets, and *T_mandatory_pscan*, see Clause 8.

Table 475—Page_Scan_Mode parameter

Value	Parameter description
0x00	Mandatory page scan mode
0x01	Optional page scan mode I
0x02	Optional page scan mode II
0x03	Optional page scan mode III
0x04–0xFF	Reserved

11.8.2 Read Page Scan Mode command

This command is used to read the default Page_Scan_Mode configuration parameter of the local device. See Table 476 and Table 477. See also 11.8.1.

Table 476—Page Scan Mode command

Command	OGF	OCF	Command parameters	Return parameters
HCI_Read_Page_Scan_Mode	0x03	0x003D		Status, Page_Scan_Mode

Table 477—Page Scan Mode return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Page Scan Mode command succeeded.
		0x01–0xFF	Read Page Scan Mode command failed. See 10.3 for list of error codes.
Page_Scan_ Mode	1 octet		See Annex B.

When the Read Page Scan Mode command has completed, a Command Complete event will be generated.

11.8.3 Write Page Scan Mode command

This command is used to write the default Page_Scan_Mode configuration parameter of the local device. See Table 478, Table 479, and Table 480. See also 11.8.1

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 478—Write Page Scan Mode command

Command	OGF	OCF	Command parameters	Return Parameters
HCI_Write_Page_Scan_Mode	0x03	0x003E	Page_Scan_Mode	Status

Table 479—Write Page Scan Mode command parameter

Parameter	Size	Value	Parameter description
Page_Scan_ Mode	1 octet		See Annex B.

Table 480—Write Page Scan Mode command return parameter

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Write Page Scan Mode command succeeded.
		0x01–0xFF	Write Page Scan Mode command failed. See 10.3 for list of error codes.

When the Write Page Scan Mode command has completed, a Command Complete event will be generated.

11.8.4 Read Country Code command

This command will read the value for the Country_Code return parameter. The country code defines which range of frequency band of the 2.4 GHz ISM band will be used by the device. Each country has local regulatory bodies regulating which 2.4 GHz ISM frequency ranges can be used. See Table 481 and Table 482.

Table 481—Read Country Code command

Command	OGF	OCF	Command parameters	Return parameters
HCI_Read_Country_Code	0x05	0x0007	None	Status, Country_Code

Table 482—Read Country Code command return parameters

Parameter	Size	Value	Parameter description
Status	1 octet	0x00	Read Country Code command succeeded.
		0x01–0xFF	Read Country Code command failed. See 10.3 for error codes and description.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 482—Read Country Code command return parameters *(continued)*

Parameter	Size	Value	Parameter description
Country_Code	1 octet	0x00	North America, Europe,[a] and Japan.
		0x01	France.
		0x04-FF	Reserved for future use.

[a]Except France

When the Read Country Code command has completed, a Command Complete event will be generated.

11.8.5 Add SCO Connection command

This command will cause the LM to create a SCO connection using the ACL connection specified by the Connection_Handle command parameter. This command causes the local device to create a SCO connection. The LM will determine how the new connection is established. This connection is determined by the current state of the device, its piconet, and the state of the device to be connected. The Packet_Type command parameter specifies which packet types the LM should use for the connection. The LM must use only the packet type(s) specified by the Packet_Type command parameter for sending HCI SCO data packets. Multiple packet types may be specified for the Packet_Type command parameter by performing a bitwise OR operation of the different packet types. The LM may choose which packet type is to be used from the list of acceptable packet types. A connection handle for this connection is returned in the Connection Complete event (see below). See Table 483 and Table 484.

Table 483—Add SCO Connection command

Command	OGF	OCF	Command parameters	Return parameters
HCI_Add_SCO_Connection	0x01	0x0007	Connection_Handle, Packet_Type	

Table 484—Add SCO Connection command parameters

Parameter	Size	Value	Parameter description
Connection_ Handle	2 octets	0xXXXX	Connection handle for the ACL connection being used to create an SCO connection. Range: 0x0000–0x0EFF (0x0F00–0x0FFF Reserved for future use)
Packet_Type	2 octets	0x0001	Reserved for future use.
		0x0002	Reserved for future use.
		0x0004	Reserved for future use.
		0x0008	Reserved for future use.
		0x0010	Reserved for future use.
		0x0020	**HV1**
		0x0040	**HV2**

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 484—Add SCO Connection command parameters *(continued)*

Parameter	Size	Value	Parameter description
Packet_Type *(continued)*		0x0080	**HV3**
		0x0100	Reserved for future use.
		0x0200	Reserved for future use.
		0x0400	Reserved for future use.
		0x0800	Reserved for future use.
		0x1000	Reserved for future use.
		0x2000	Reserved for future use.
		0x4000	Reserved for future use.
		0x8000	Reserved for future use.

An SCO connection can be created only when an ACL connection already exists and when it is not put in PARK state. For a definition of the different packet types, see Clause 8.

At least one packet type must be specified. The host should enable as many packet types as possible for the LM to perform efficiently. However, the host must not enable packet types that the local device does not support.

When the controller receives the Add SCO Connection command, it sends the Command Status event to the host. In addition, when the LM determines the connection is established, the local controller will send a Connection Complete event to its host, and the remote controller will send a Connection Complete event or a Synchronous Connection Complete event to the host. The Connection Complete event contains the connection handle if this command is successful.

No Command Complete event will be sent by the controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

11.8.6 Page Scan Mode Change event

The Page Scan Mode Change event indicates that the connected remote device with the specified BD_ADDR has successfully changed the page scan mode. See Table 485 and Table 486.

Table 485—Page Scan Mode Change event

Event	Event code	Event parameters
Page Scan Mode Change	0x1F	BD_ADDR, Page_Scan_Mode

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 486—Page Scan Mode Change event parameters

Parameter	Size	Value	Parameter description
BD_ADDR	6 octets	0xXXXXXX XXXXXX	BD_ADDR of the remote device.
Page_Scan_Mode	1 octet	0x00	Mandatory page scan mode.
		0x01	Optional page scan mode I.
		0x02	Optional page scan mode II.
		0x03	Optional page scan mode III.
		0x04–0xFF	Reserved.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12. Message sequence charts (MSCs)

This clause has examples of interactions between HCI commands and events and LMP data units are represented in the form of MSCs. These charts show typical interactions and do not indicate all possible protocol behavior.

12.1 Overview

This clause shows typical interactions between HCI commands and events and LMP PDUs. It focuses on the MSCs for the procedures specified in Clause 11 with regard to LM procedures from Clause 9.

This clause illustrates only the most useful scenarios; it does not cover all possible alternatives. Furthermore, the MSCs do not consider errors over the air interface or host interface. In all MSCs, it is assumed that all events are not masked so the host controller will not filter out any events.

The sequence of messages in these MSCs is for illustrative purposes. The messages may be sent in a different order where allowed by Clause 9 or Clause 11. If any of these charts differs with text in Clause 8, Clause 9, or Clause 11, the text in those clauses takes precedence.

12.1.1 Notation

The notation used in the MSCs consists of ovals, elongated hexagons, boxes, lines, and arrows. The vertical lines terminated on the top by a shadow box and at the bottom by solid oval indicate a protocol entity that resides in a device. MSCs describe interactions between these entities and states those entities may be in.

The symbols in Table 487 represent interactions and states.

Table 487—MSC symbols

Shape	Function
Oval	Defines the context for the MSC.
Hexagon	Indicates a condition needed to start the transactions below this hexagon. The location and width of the hexagon indicate which entity or entities make this decision.
Box	Replaces a group of transactions. May indicate a user action or a procedure in the BB.
Dashed box	Optional group of transactions.
Solid arrow	Represents a message, signal, or transaction. Can be used to show LMP and HCI traffic. Some BB packet traffic is also shown. These are prefixed by BB followed by either the type of packet or an indication that there is an ACK signal in a packet.
Dashed arrow	Represents a optional message, signal, or transaction. Can be used to show LMP and HCI traffic.

12.1.2 Flow of control

Some message sequences are split into several charts. These charts are marked in sequence with different step numbers, and then the alternative paths are marked with different letters after the step numbers. Numbers indicate normal or required ordering. The letters represent alternative paths. For example, Step 4 is after Step 3, and Step 5a could be executed instead of Step 5b.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.1.3 Sample MSC

The protocol entities represented in the example shown in Figure 99 illustrate the interactions of two devices named A and B. Note that each device includes a host and a LM entity in this example. Other MSCs in this subclause may show the interactions of more than two devices.

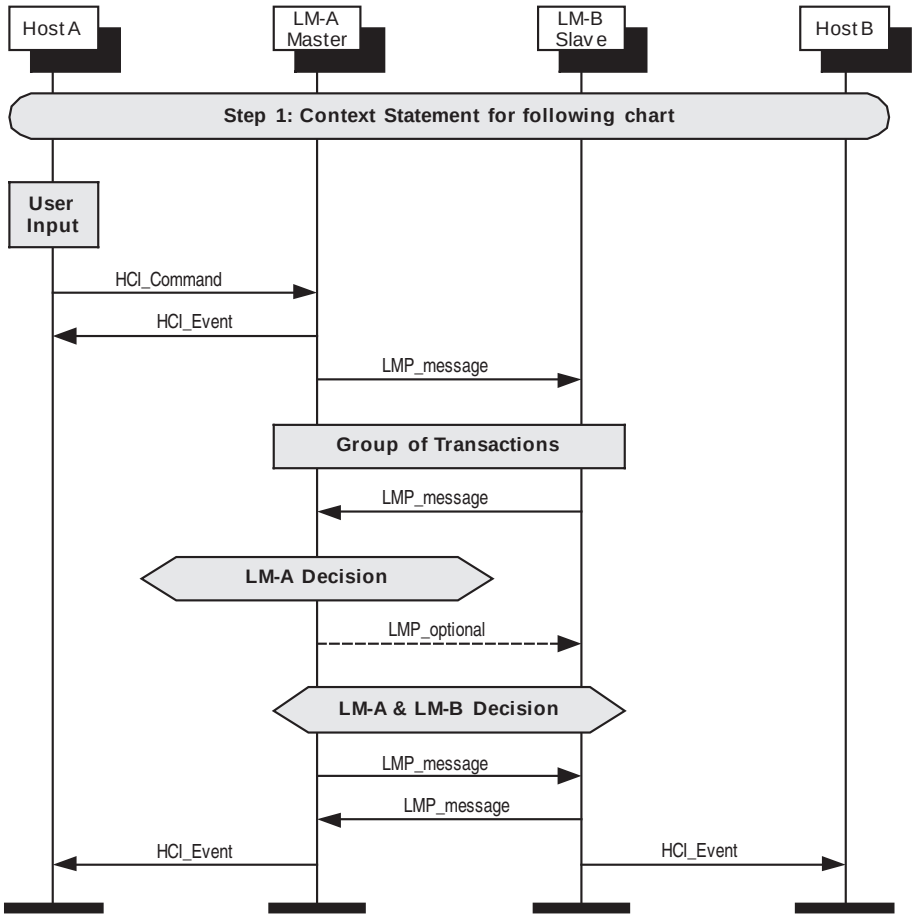


Figure 99—Sample MSC

12.2 Services without connection request

12.2.1 Remote name request

The remote name request service is used to find out the name of the remote device without requiring an explicit ACL connection.

Step 1: The host sends an HCI_Remote_Name_Request command expecting that its local device will automatically try to connect to the remote device (see Figure 100).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

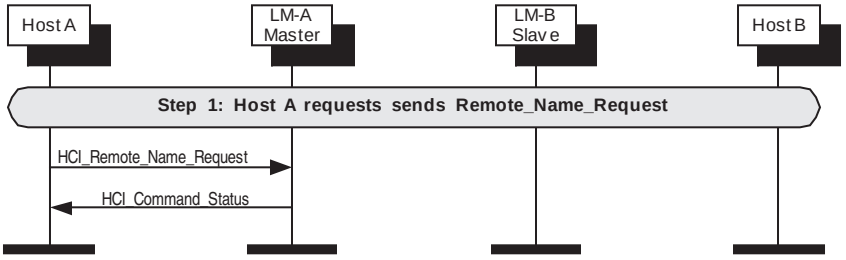


Figure 100—Remote name request

Step 2a: If an ACL connection does not exist, device A pages device B. After the BB paging procedure, the local device attempts to get the name, disconnect, and return the name of the remote device to the host (see Figure 101).

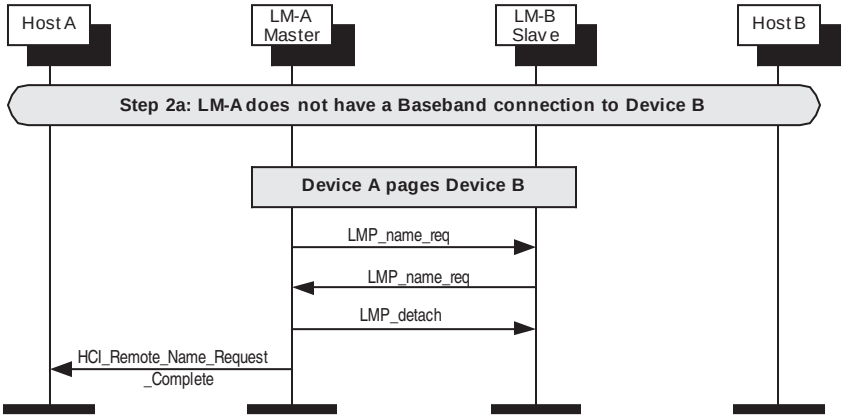


Figure 101—Remote name request if no current BB connection

Step 2b: If an ACL connection exists when the request is made, then the remote name request procedure will be executed like an optional service. No paging and no ACL disconnect is done (see Figure 102).

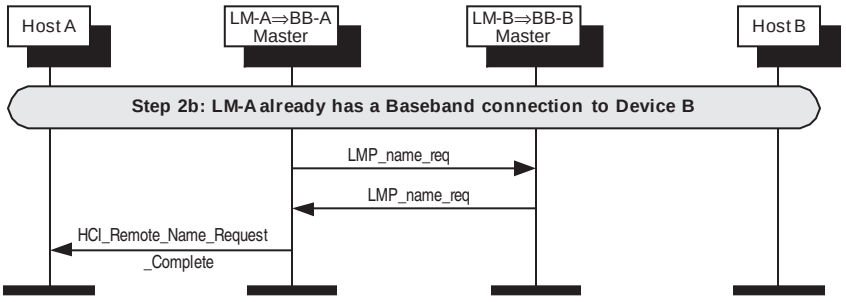


Figure 102—Remote name request with BB connection

12.2.2 One-time inquiry

Inquiry is used to detect and collect nearby devices.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1: The host sends an HCI_Inquiry command (see Figure 103).

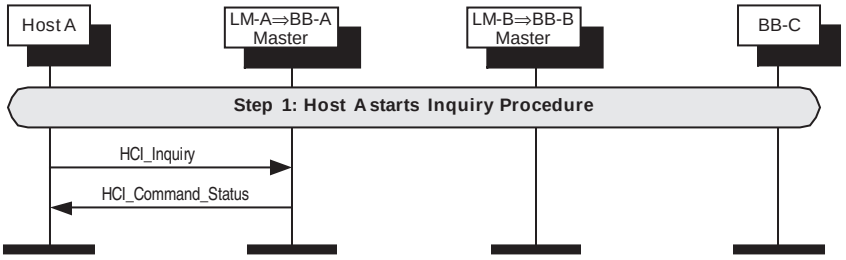


Figure 103—Host A starts inquiry procedure

Step 2: The controller will start the BB inquiry procedure with the specified IAC and Inquiry length. When inquiry responses are received, the controller extracts the required information and returns the information related to the found devices using one or more Inquiry Result events to the host (see Figure 104).

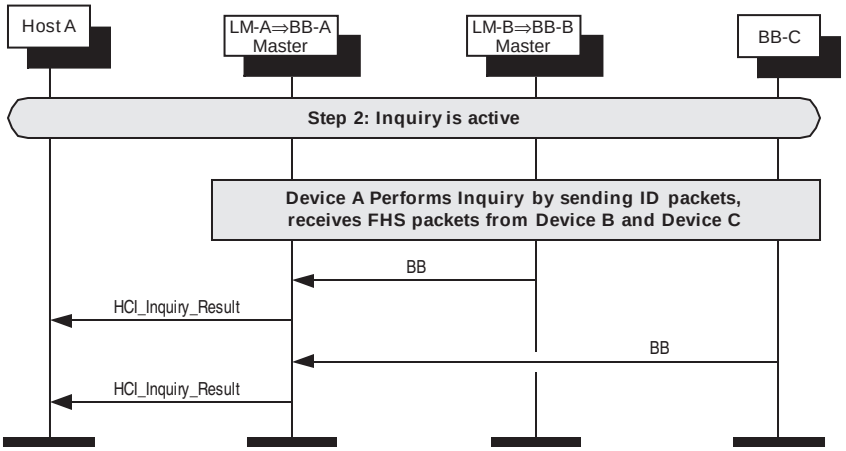


Figure 104—LM-A performs inquiry and reports result

Step 3a: If the host wishes to terminate an inquiry, the HCI_Inquiry_Cancel command is used to immediately stop the inquiry procedure (see Figure 105).

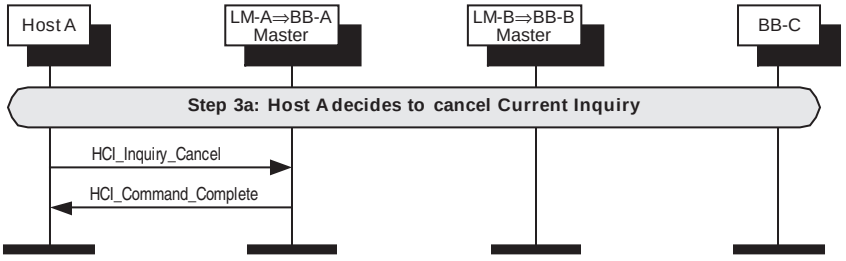


Figure 105—Host A cancels inquiry

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 3b: If the inquiry procedure is completed due to the number of results obtained or the inquiry length has expired, an Inquiry Complete event is returned to the host (see Figure 106).

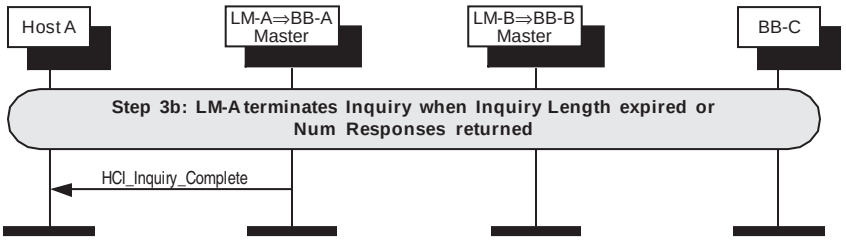


Figure 106—LM-A terminates current inquiry

12.2.3 Periodic inquiry

Periodic inquiry is used when the inquiry procedure is to be repeated periodically.

Step 1: The hosts sends an HCI_Periodic_Inquiry_Mode command (see Figure 107).

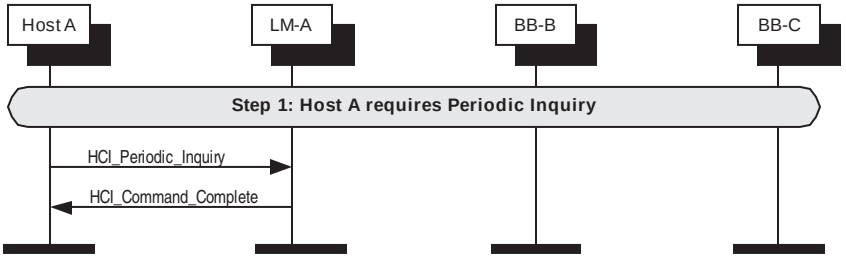


Figure 107—Host A starts periodic inquiry

Step 2: The controller will start a periodic inquiry. In the inquiry cycle, one or several Inquiry Result events will be returned (see Figure 108).

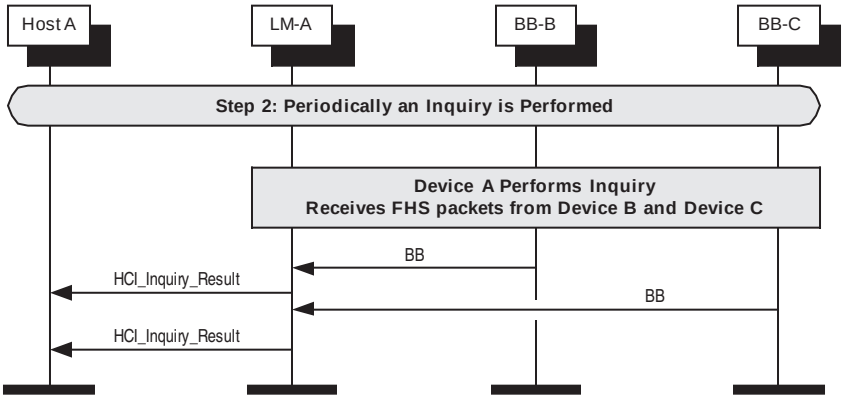


Figure 108—LM-A periodically performs an inquiry and reports result

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 3: An Inquiry Complete event will be returned to the host when the current periodic inquiry has finished (see Figure 109).

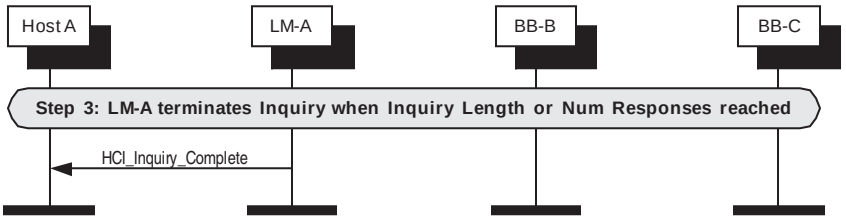


Figure 109—LM-A terminates current inquiry

Step 4: The periodic inquiry can be stopped using the HCI_Exit_Periodic_Inquiry_Mode command (see Figure 110).

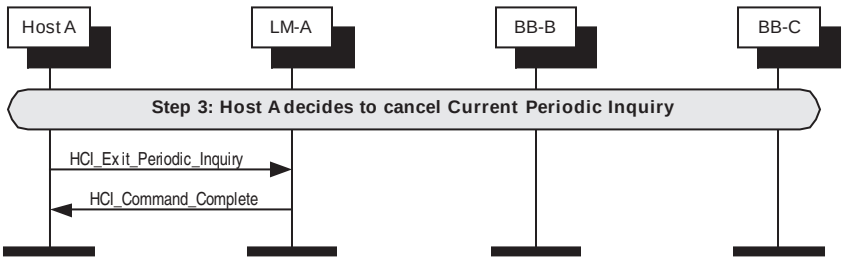


Figure 110—Host A decides to exit periodic inquiry

12.3 ACL Connection establishment and detachment

A flow diagram of the establishment and detachment of a connection between two devices is shown in Figure 111. The process is illustrated in nine distinct steps. A number of these steps may be optionally performed, such as authentication and encryption. Some steps are required, such as the connection request and setup complete steps. The steps in the overview diagram directly relate to the steps in the MSCs in this subclause.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

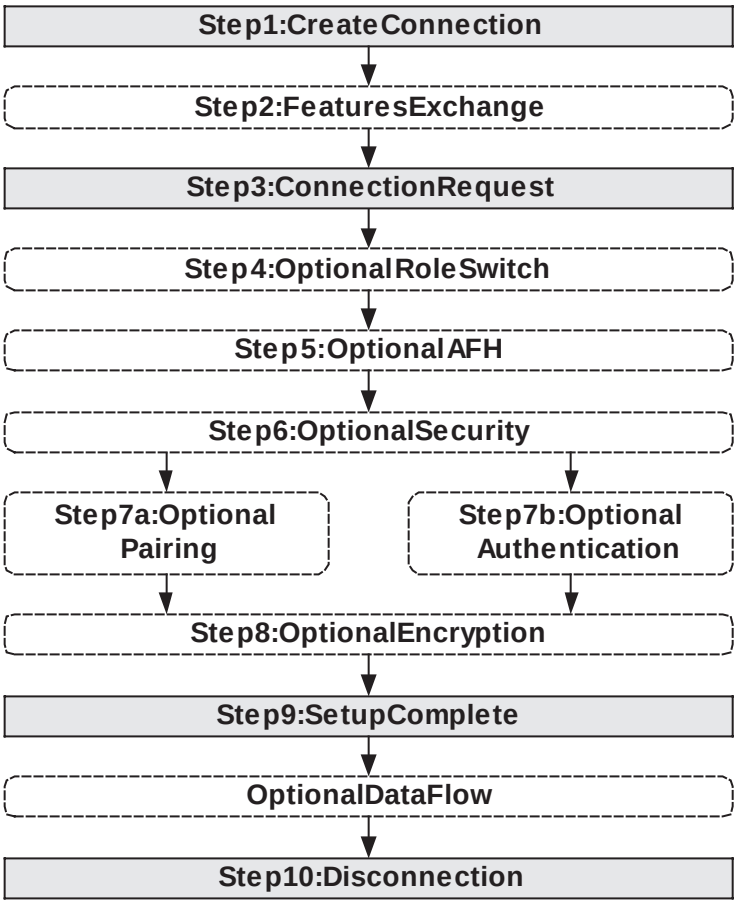


Figure 111—Overview diagram for connection setup

12.3.1 Connection setup

Step 1: The host sends an HCI_Create_Connection command to the controller. The controller then performs a BB paging procedure with the specified BD_ADDR (see Figure 112).

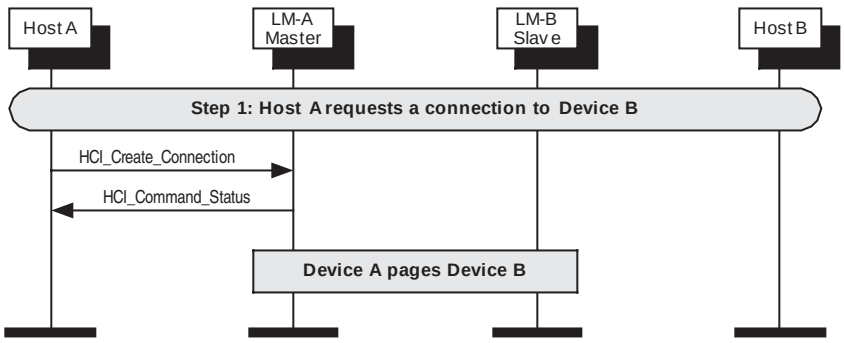


Figure 112—Host A requests connection with device B

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2: Optionally, the LM may decide to exchange features (see Figure 113).

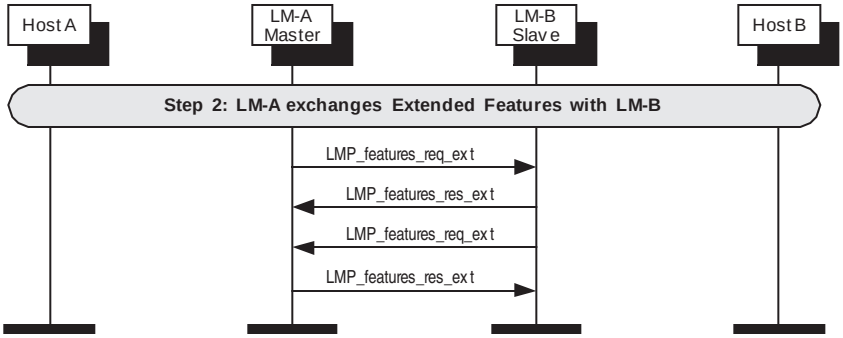


Figure 113—LM-A and LM-B exchange features

Step 3: The LM on the master will request an LMP_host_connection_req PDU. The LM on the slave will then confirm that a connection is OK, and if so, what role is preferred (see Figure 114).

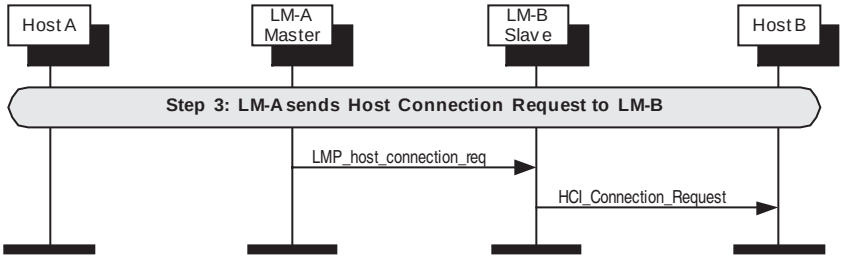


Figure 114—LM-A requests host connection

Step 4a: The remote host rejects this connection, and the link is terminated (see Figure 115).

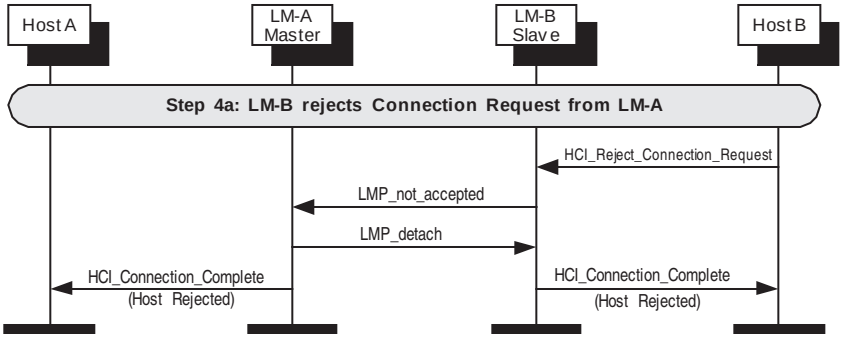


Figure 115—Device B rejects connection request

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 4b: The remote host accepts this connection (see Figure 116).

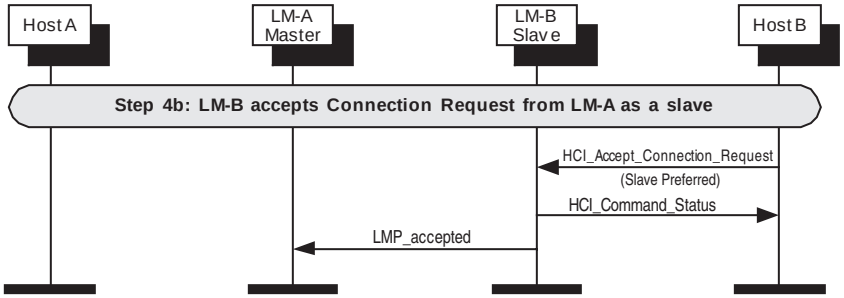


Figure 116—Device B accepts connection request

Step 4c: The remote host accepts this connection, but with the preference of being a master. This will cause a role switch to occur before the LMP_accepted PDU for the LMP_host_connection_req PDU is sent (see Figure 117).

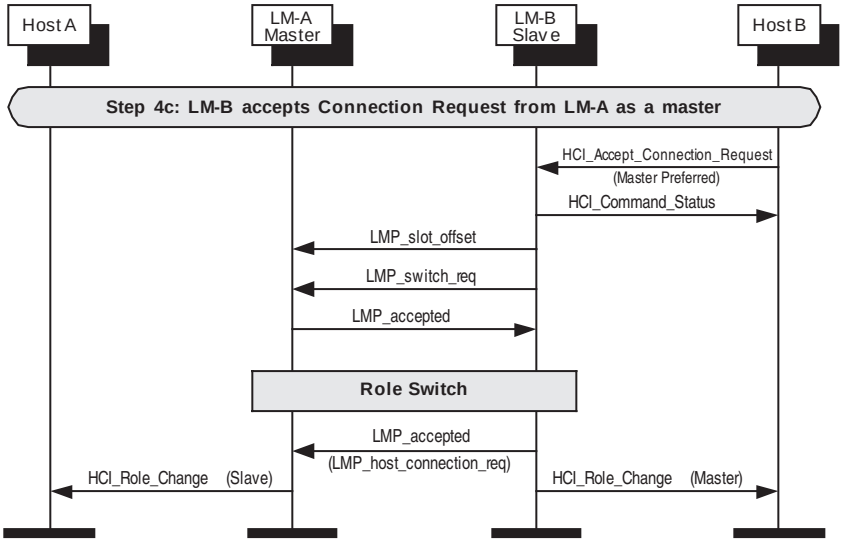


Figure 117—Device B accepts connection requests as master

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 5: After the features have been exchanged and AFH support is determined to be available, the master may at any time send an LMP_set_AFH PDU and an LMP_channel_classification_req PDU (see Figure 118).

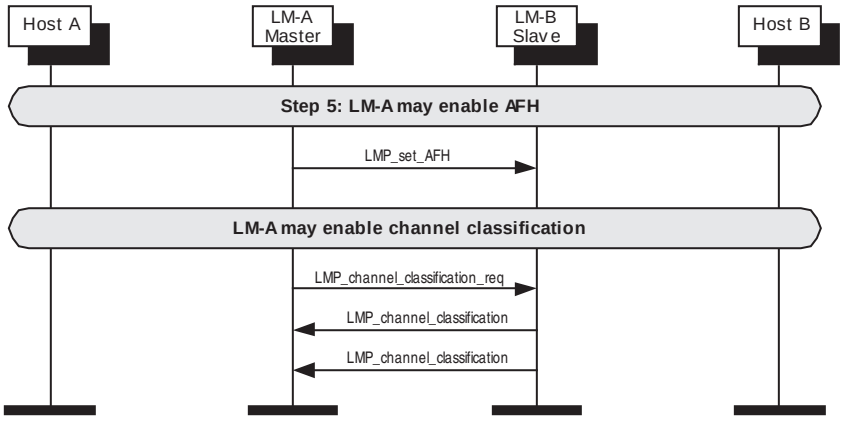


Figure 118—LM-A starts AFH

Step 6: The LM will request if authentication is required. It does this by requesting the link key for this connection from the host (see Figure 119).

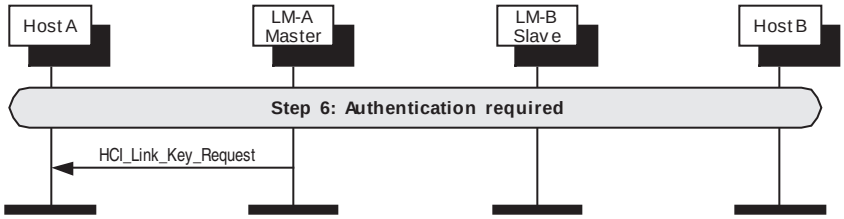


Figure 119—Authentication initiated

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 7a: If authentication is required by the higher layers and the devices to be connected do not have a common link key, a pairing procedure will be used. The LM will have requested a link key from the host for this connection. If there is a negative reply, then a PIN code will be requested. This PIN code will be requested on both sides of the connection, and authentication will be performed based on this PIN code. The last step is for the new link key for this connection to be passed to the host so that it may store it for future connections (see Figure 120).

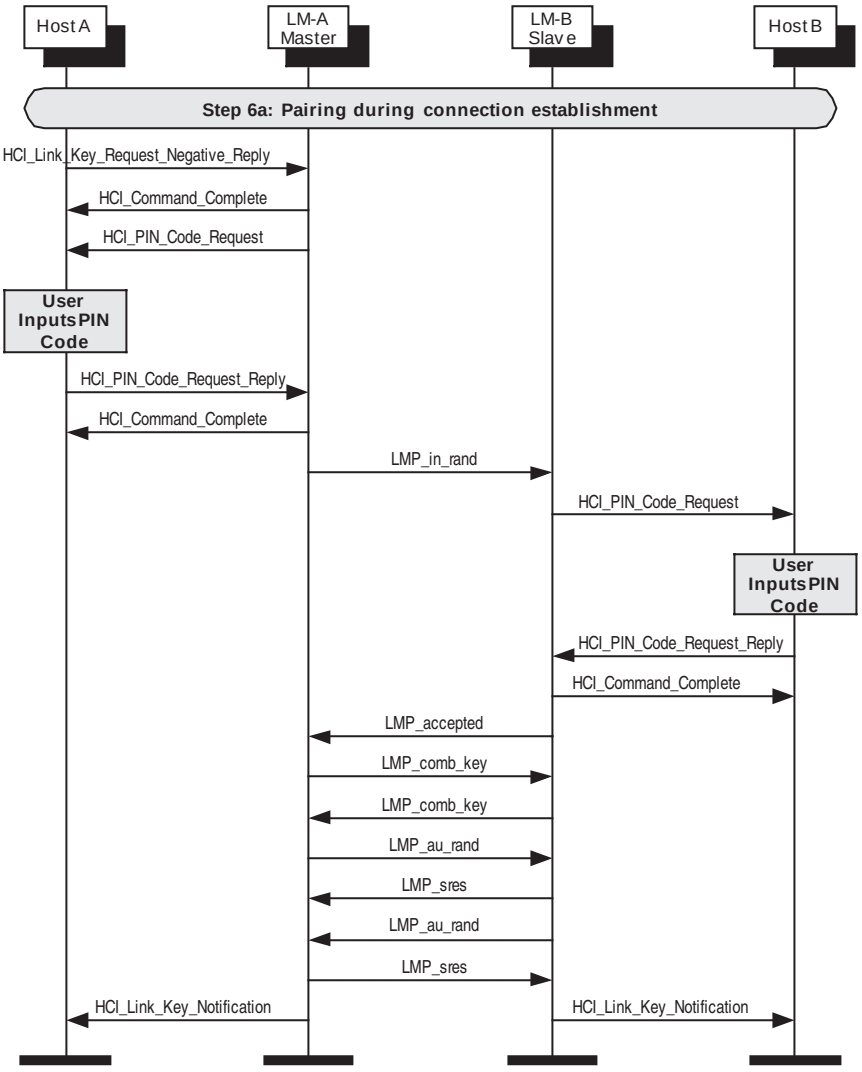


Figure 120—Pairing during connection setup

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 7b: If a common link key exists between the devices, then pairing is not needed. The LM will have asked for a link key from the host for this connection. If this is a positive reply, then the link key is used for authentication. If the Authentication_Enable configuration parameter is set, then the authentication procedure must be executed. This MSC shows only the case when the Authentication_Enable configuration parameter is set on both sides (see Figure 121).

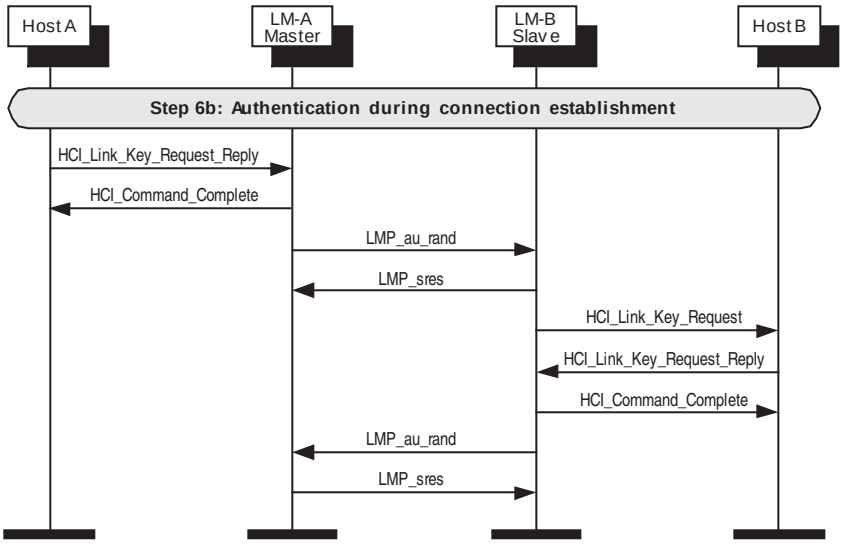


Figure 121—Authentication during connection setup

Step 8: Once the pairing or authentication procedure is successful, the encryption procedure may be started. This MSC shows only the setup of an encrypted point-to-point connection (see Figure 122).

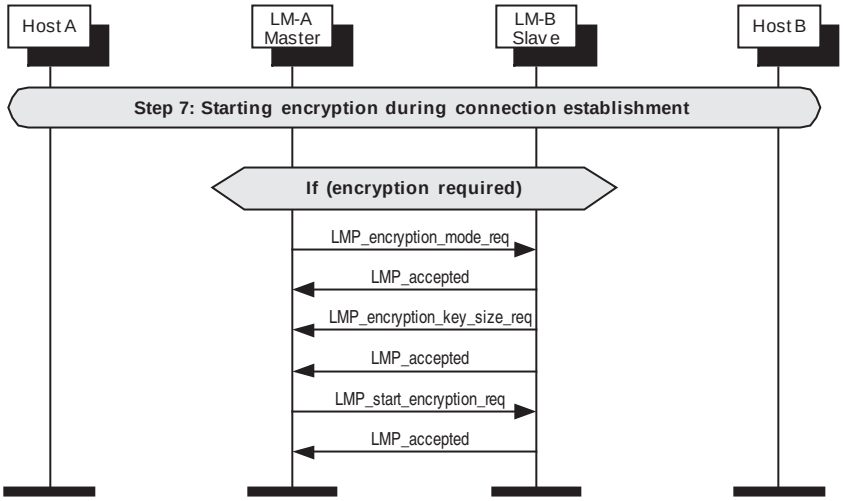


Figure 122—Starting encryption during connection setup

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 9: The LMs indicate that the connection is set up by sending LMP_setup_complete PDU. This will cause the host to be notified of the new connection handle, and this connection may be used to send higher layer data such as L2CAP information (see Figure 123).

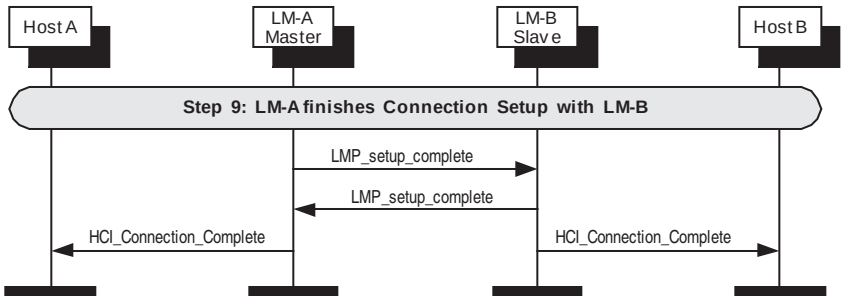


Figure 123—LM-A and LM-B finish connection setup

Step 10: Once the connection is no longer needed, either device may terminate the connection using the HCI_Disconnect command and LMP_detach PDU. The disconnection procedure is one-sided and does not need an explicit acknowledgment from the remote LM. The use of ARQ acknowledgment from the BB is needed to ensure that the remote LM has received the LMP_detach PDU (see Figure 124).

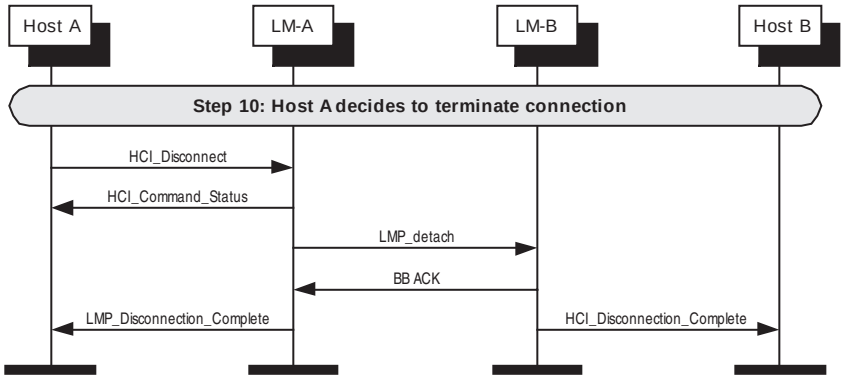


Figure 124—Host A decides to disconnect

12.4 Optional activities after ACL connection establishment

12.4.1 Authentication requested

Step 1: Authentication can be explicitly executed at any time after a connection has been established. If no link key is available, then the link key is required from the host (see Figure 125).

Note: If the controller or LM and the host do not have the link key, a PIN Code Request event will be sent to the host to request a PIN code for pairing. A procedure identical to that used during connection setup (see 12.3.1, Step 7a:) will be used (see Figure 119).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

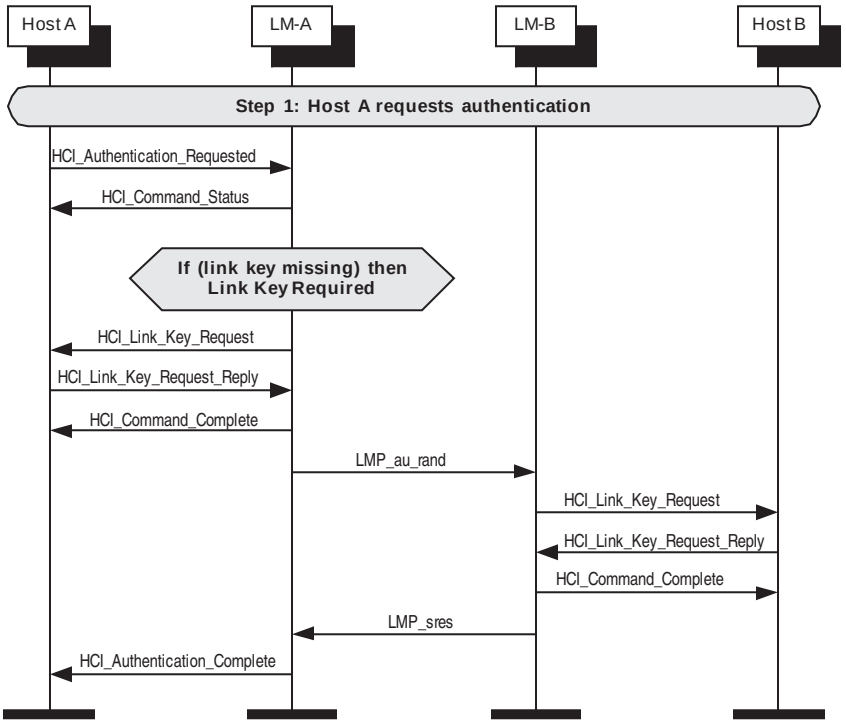


Figure 125—Authentication requested

12.4.2 Set connection encryption

Step 1: The host may at any time turn on encryption using the HCI_Set_Connection_Encryption command. This command can be originated from either the master or slave sides. Only the master side is shown in Figure 126. If this command is sent from a slave, the only difference is that the LMP_encryption_mode_req PDU will be sent from the slave. The LMP_encryption_key_size_req and LMP_start_encryption_req PDUs will always be requested from the master (see Figure 126).

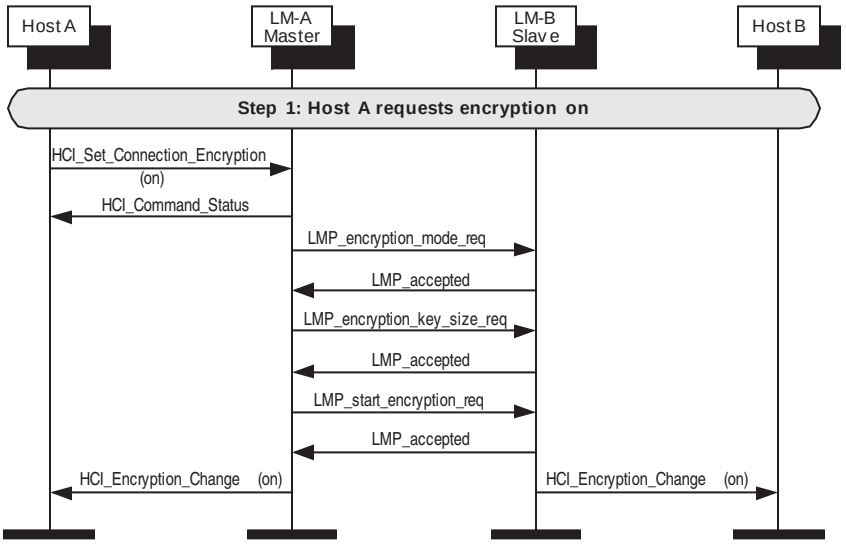


Figure 126—Encryption requested

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2: To terminate the use of encryption, the HCI_Set_Connection_Encryption command is used (see Figure 127).

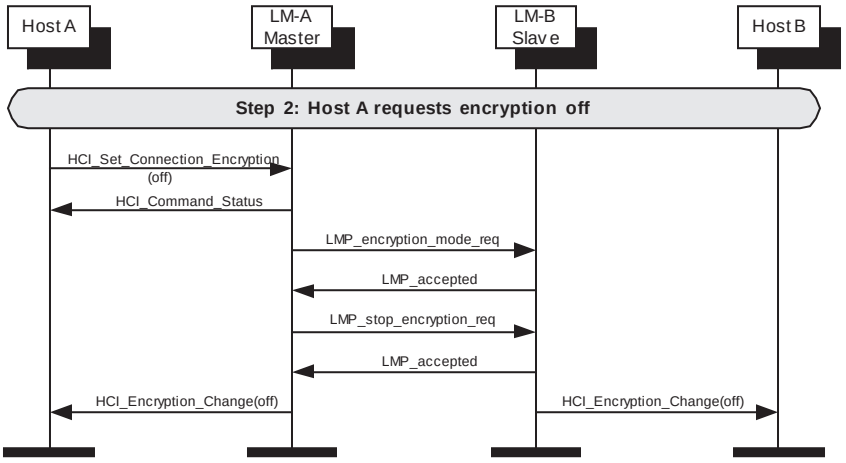


Figure 127—Encryption off requested

12.4.3 Change connection link key

Step 1: The master host (host A) may change the connection link key using the HCI_Change_ Connection_Link_Key command. A new link key will be generated, and the hosts will be notified of this new link key (see Figure 128).

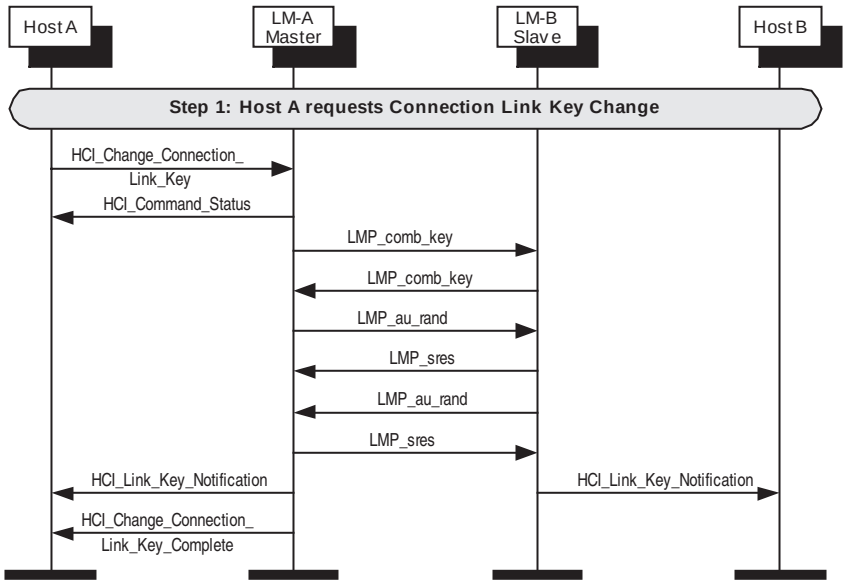


Figure 128—Change connection link key

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.4.4 Master link key

Step 1: The host changes to a master link key from a semi-permanent link key using the HCI_Master_Link_Key command (see Figure 129).

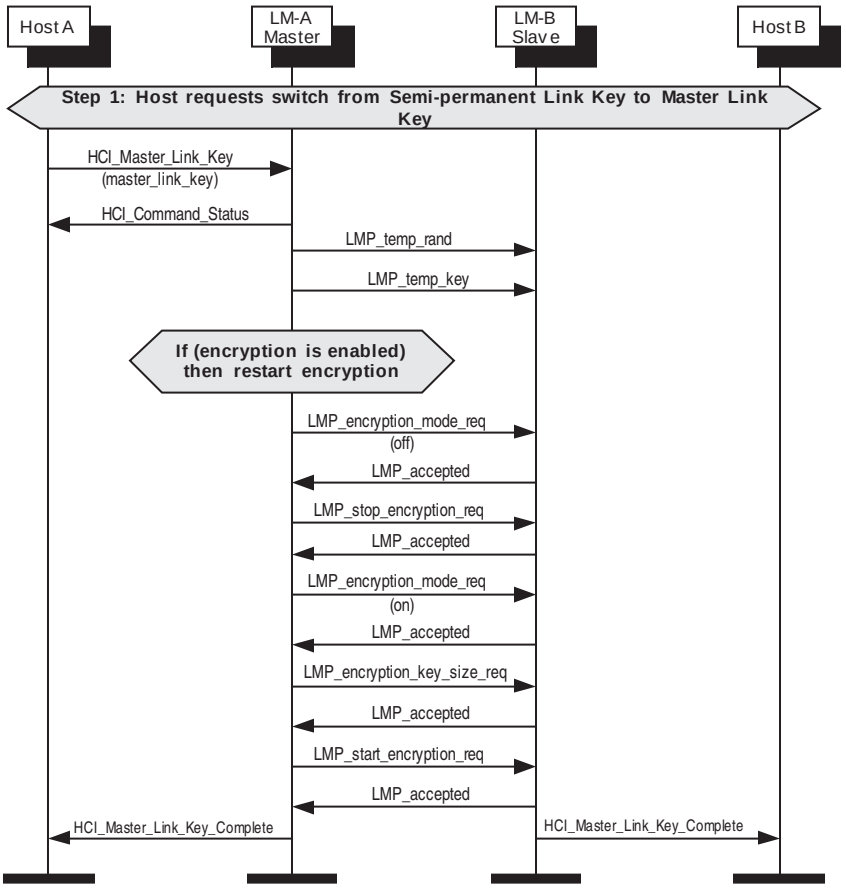


Figure 129—Change to master link key

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2: The host changes to a semi-permanent link key from a master link key using the HCI_Master_Link_Key command (see Figure 130).

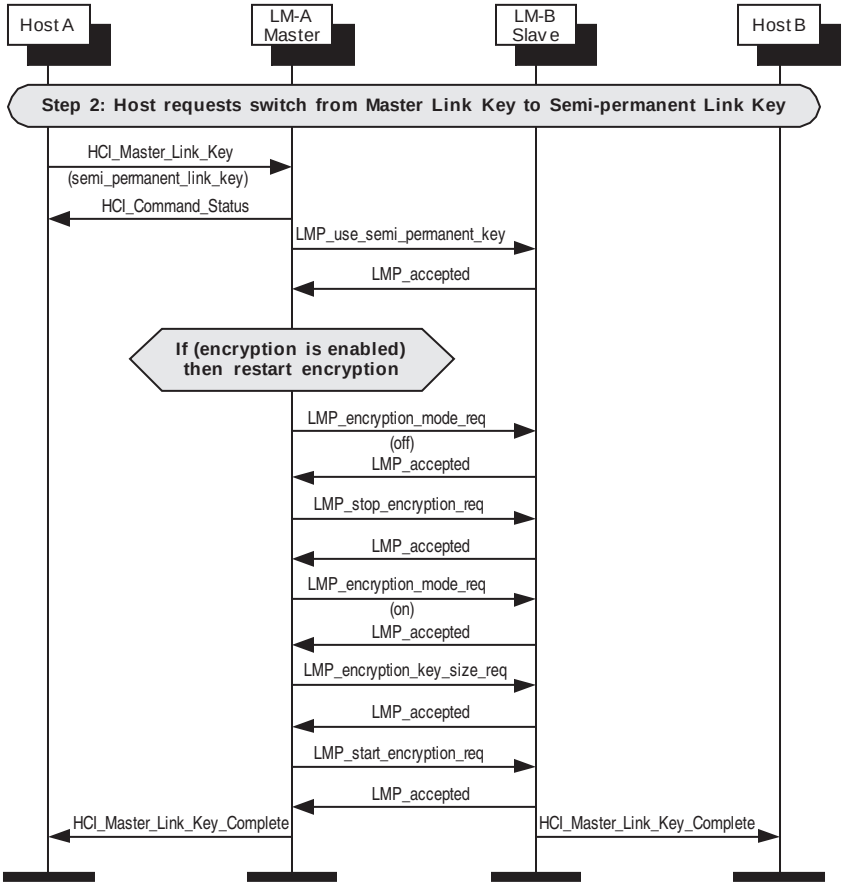


Figure 130—Change to semi permanent link key

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.4.5 Read remote supported features

Using the HCI_Read_Remote_Supported_Features command, the supported LMP features of a remote device can be read (see Figure 131).

If the remote supported features have been obtained previously, then the controller may return them without sending any LMP PDUs.

Step 1: The host requests the supported features of a remote device.

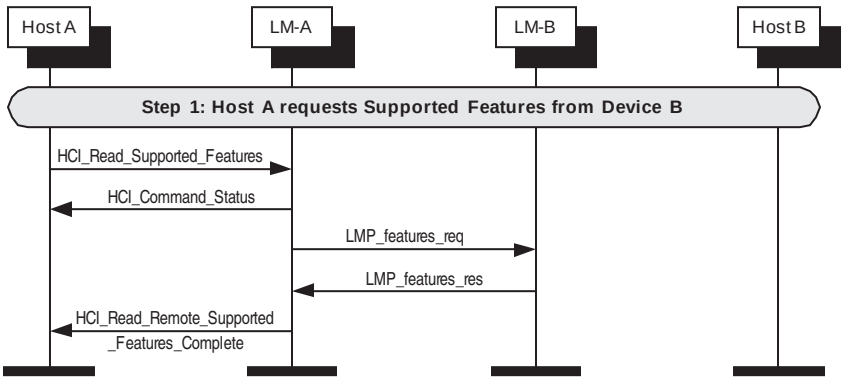


Figure 131—Read remote supported features

12.4.6 Read remote extended features

Using the HCI_Read_Remote_Extended_Features command, the extended LMP features of a remote device can be read (see Figure 132).

If the remote extended features have been obtained previously, then the controller may return them without sending any LMP PDUs.

Step 1: The host requests the extended features of a remote device.

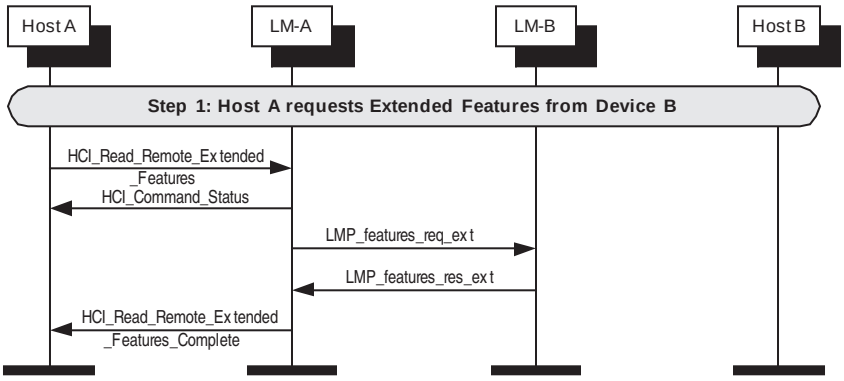


Figure 132—Read remote extended features

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.4.7 Read clock offset

Using the HCI_Read_Clock_Offset command, the device acting as the master can read the clock offset of a slave. The clock offset can be used to speed up the paging procedure in a later connection attempt. If the command is requested from the slave device, the controller will directly return a Command Status event and a Read Clock Offset Complete event without sending any LMP PDUs (see Figure 133).

Step 1: The host requests the clock offset of a remote device.

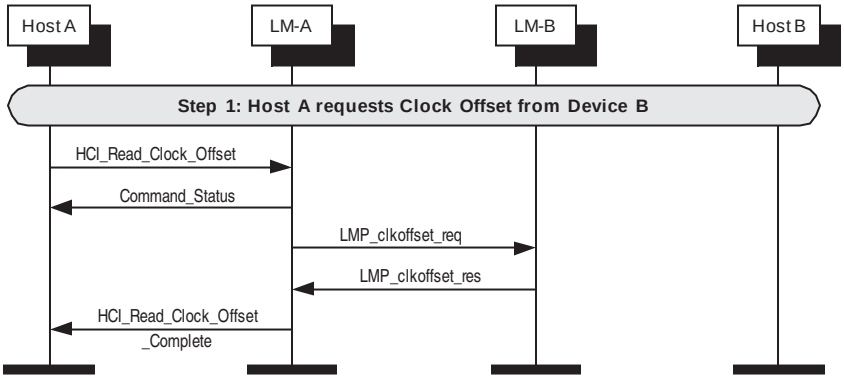


Figure 133—Read clock offset

12.4.8 Read remote version information

Using the HCI_Read_Remote_Version_Information command, the version information of a remote device can be read (see Figure 134).

If the remote version information has been obtained previously, then the controller may return them without sending any LMP PDUs.

Step 1: The host requests the version information of a remote device.

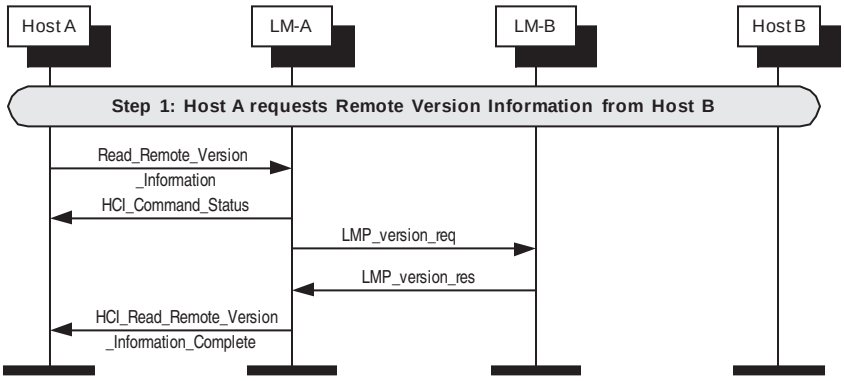


Figure 134—Read remote version information

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.4.9 QoS setup

Using the HCI_Flow_Specification command, the QoS and flow specification requirements of a connection can be notified to a controller. The controller may then change the QoS parameters with a remote device (see Figure 135).

Step 1: The host sends QoS parameters to a remote device.

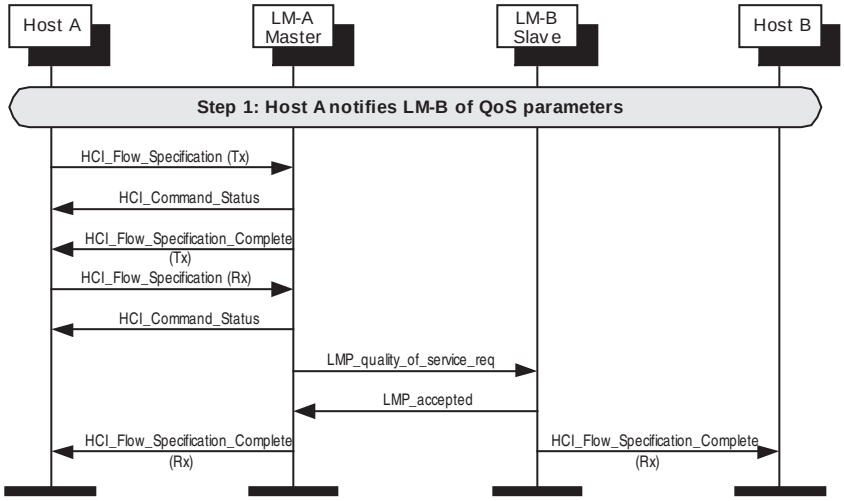


Figure 135—QoS flow specification

12.4.10 Switch role

The HCI_Switch_Role command can be used to explicitly switch the current master-slave role of the local device with the specified device.

Step 1a: The master host (A) requests a role switch with a slave. This will send the switch request, and the slave will respond with the slot offset and LMP_accepted PDU (see Figure 136).

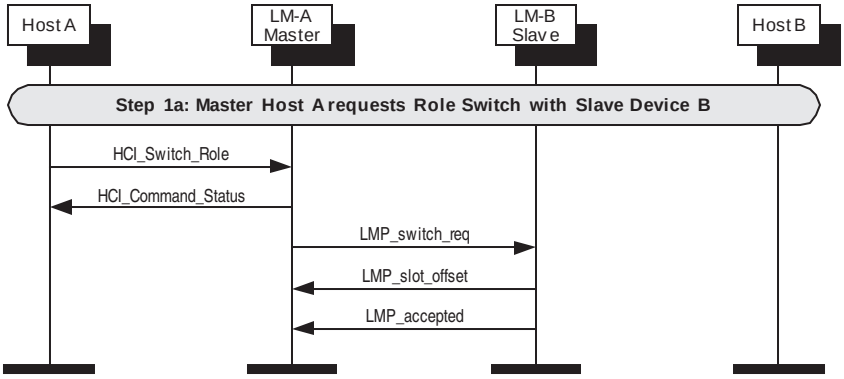


Figure 136—Master requests role switch

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1b: The slave host (B) requests a role switch with a master. This will send the slot offset and the switch request, and the master will respond with a LMP_accepted PDU (see Figure 137).

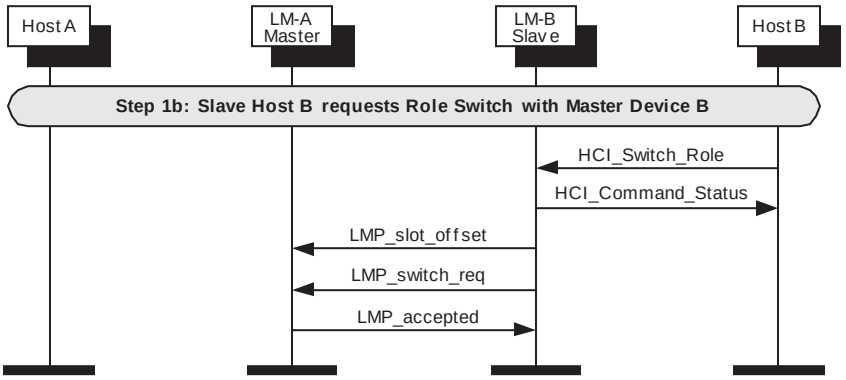


Figure 137—Slave requests role switch

Step 2: The role switch is performed by doing the TDD switch and piconet switch. Finally an HCI_Role_Change event is sent on both sides (see Figure 138).

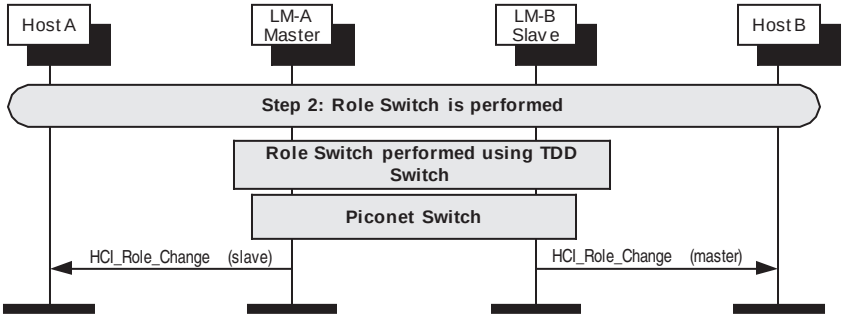


Figure 138—Role switch is performed

12.5 Synchronous connection establishment and detachment

12.5.1 Synchronous connection setup

Using the HCI_Setup_Synchronous_Connection command, a host can add a synchronous logical channel to the link. A synchronous logical link can be provided by creating a SCO or an eSCO logical transport.

NOTE—An ACL connection must be established before a synchonous connection can be created.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1a: Master device requests a synchronous connection with a device (see Figure 139).

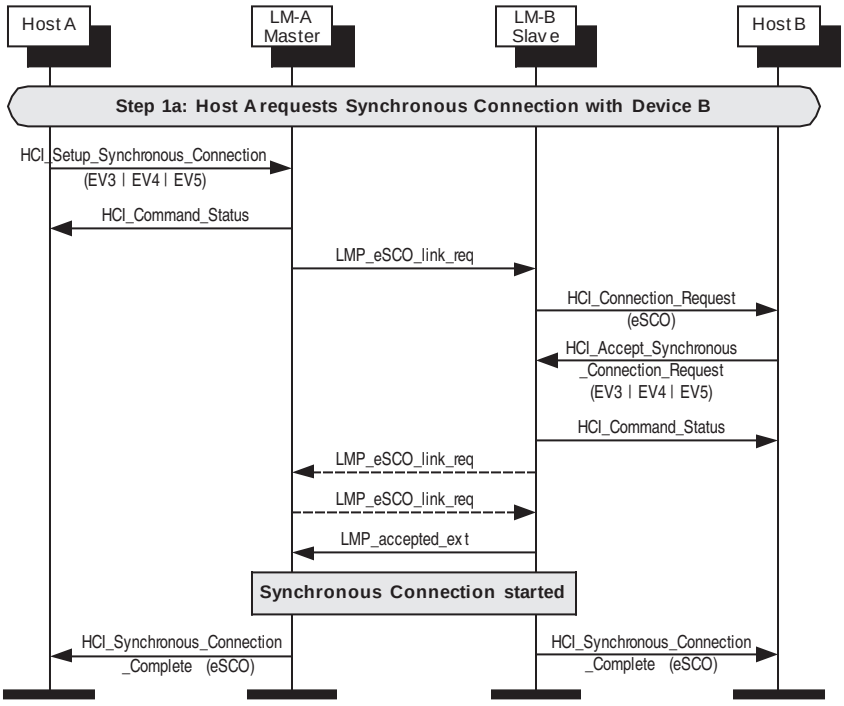


Figure 139—Master requests synchronous EV3, EV4, OR EV5 connection

Step 1b: Slave device requests a synchronous connection with a device (see Figure 140).

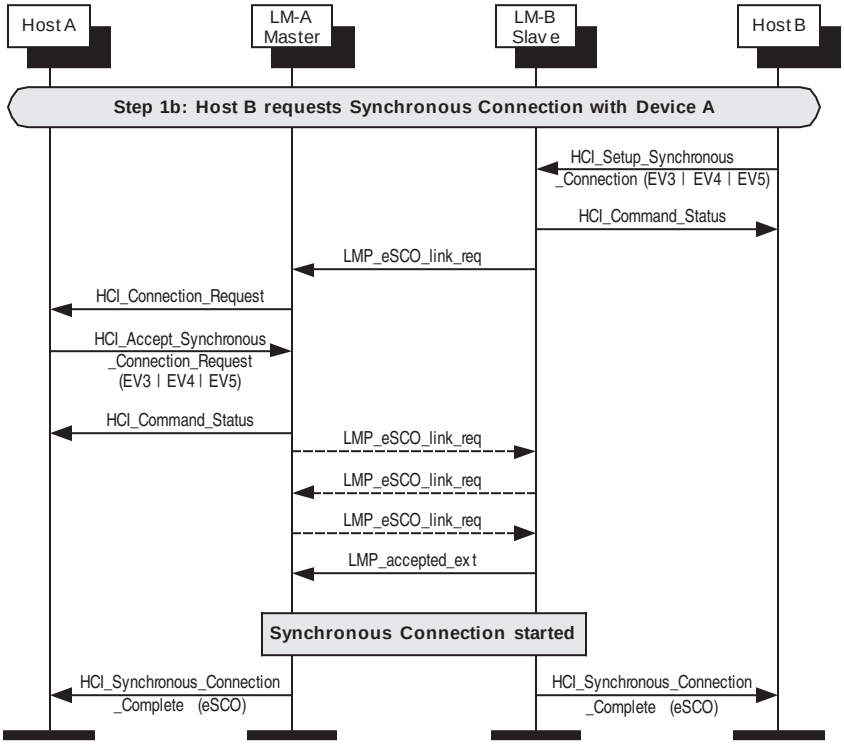


Figure 140—Slave requests synchronous EV3, EV4, OR EV5 connection

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1c: Master device requests a SCO connection with a device (see Figure 141).

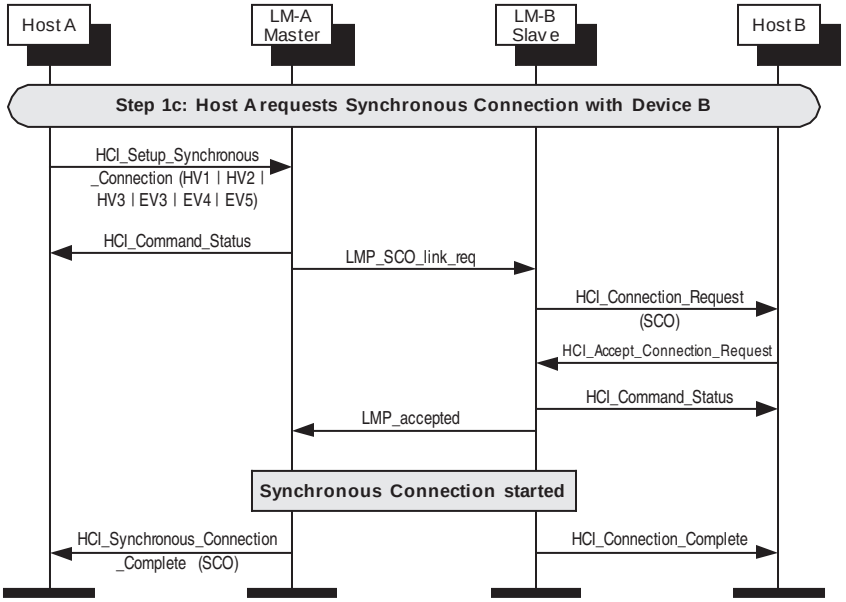


Figure 141—Master requests synchronous connection using SCO

Step 1d: Master device requests a SCO connection with a device (see Figure 142).

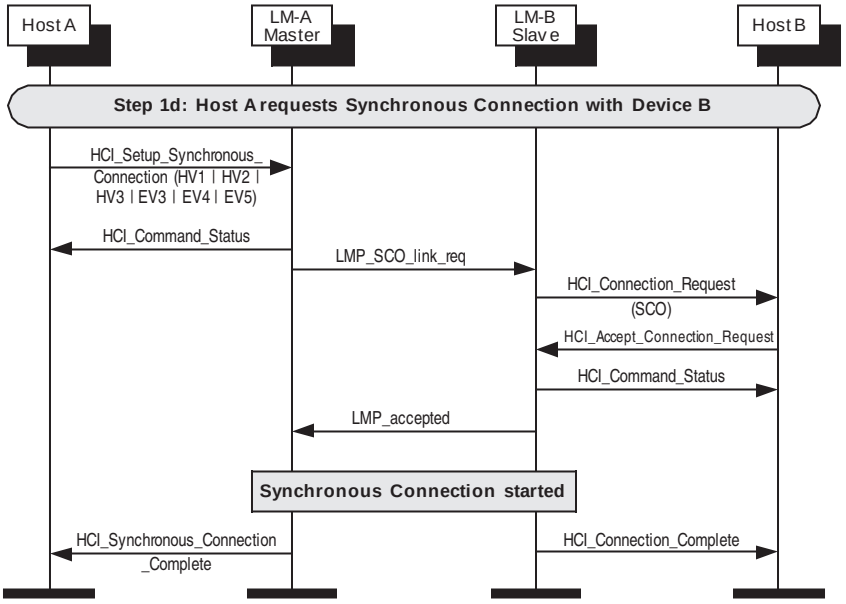


Figure 142—Master requests synchronous connection with legacy slave

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1e: Host device requests a SCO connection with a device (see Figure 143).

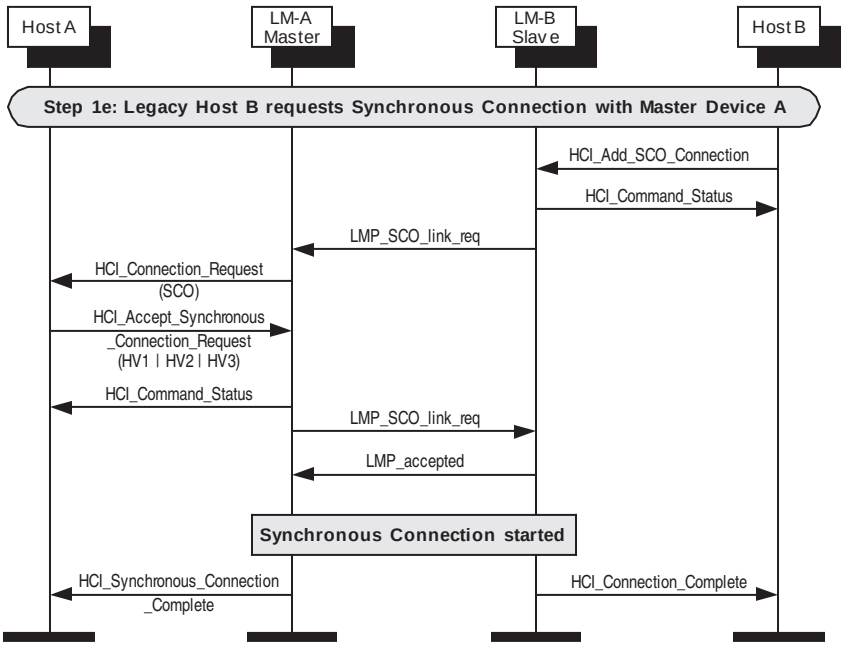


Figure 143—Any device that supports only SCO connections requests a synchronous connection with a device

Step 2a: Master renegotiates eSCO connection (see Figure 144).

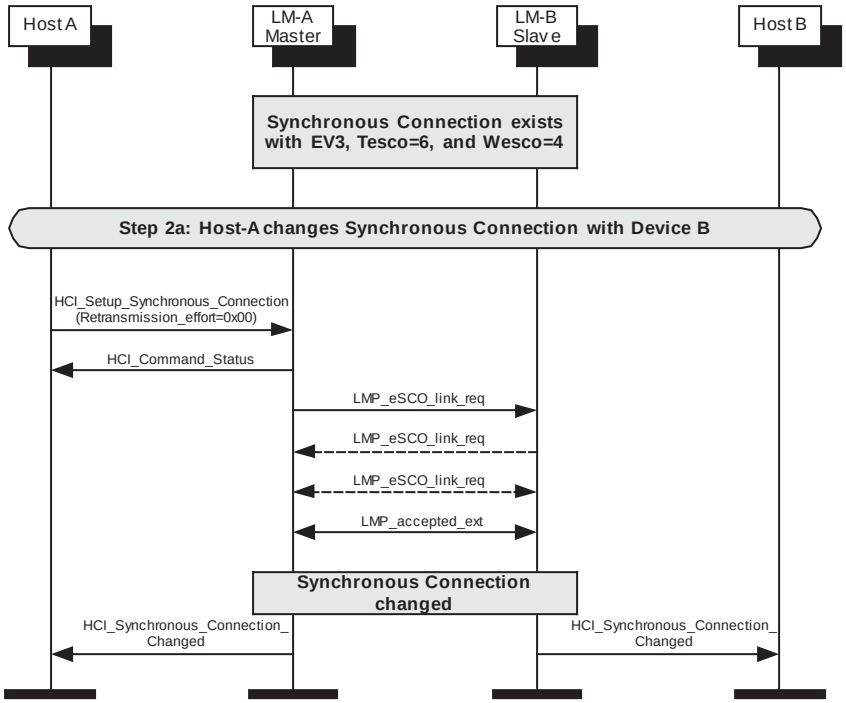


Figure 144—Master renegotiates eSCO connection

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2b: Slave renegotiates eSCO connection (see Figure 145).

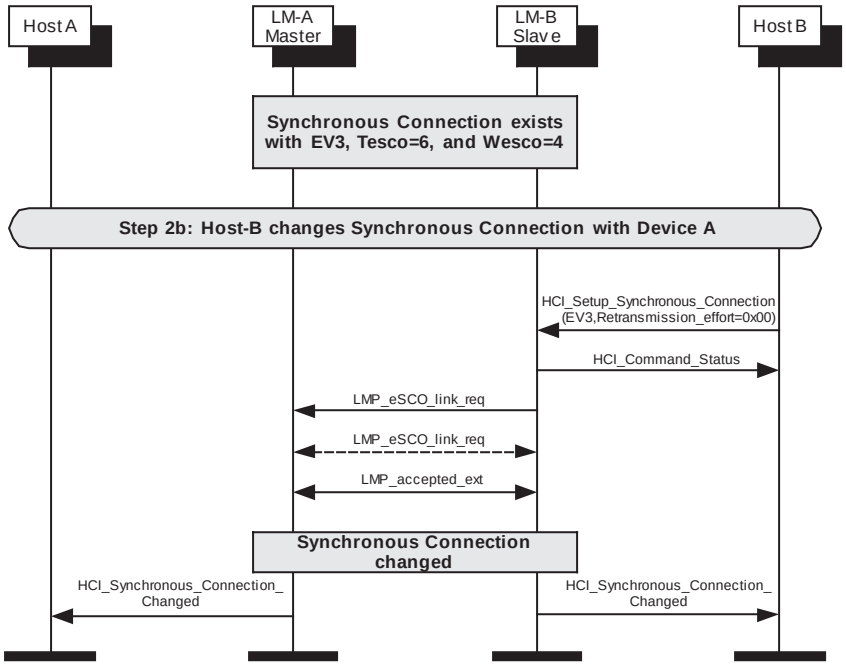


Figure 145—Slave renegotiates eSCO connection

Step 3a: eSCO disconnection (see Figure 146).

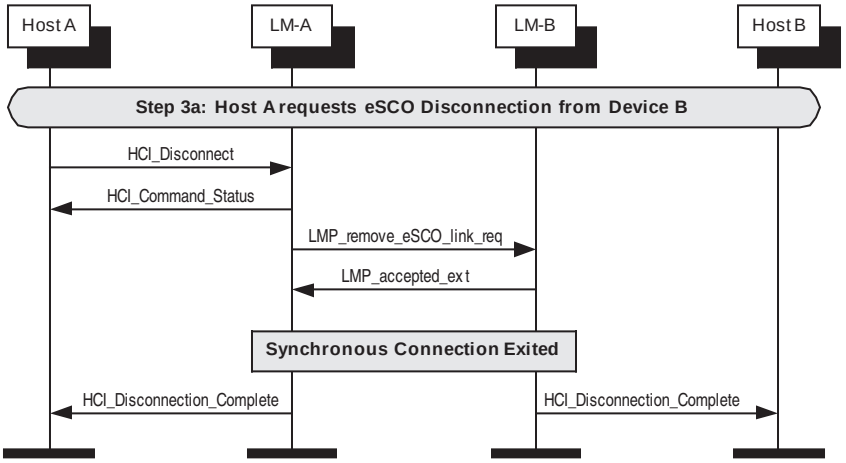


Figure 146—Synchronous disconnection of eSCO connection

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 3b: SCO disconnection (see Figure 147).

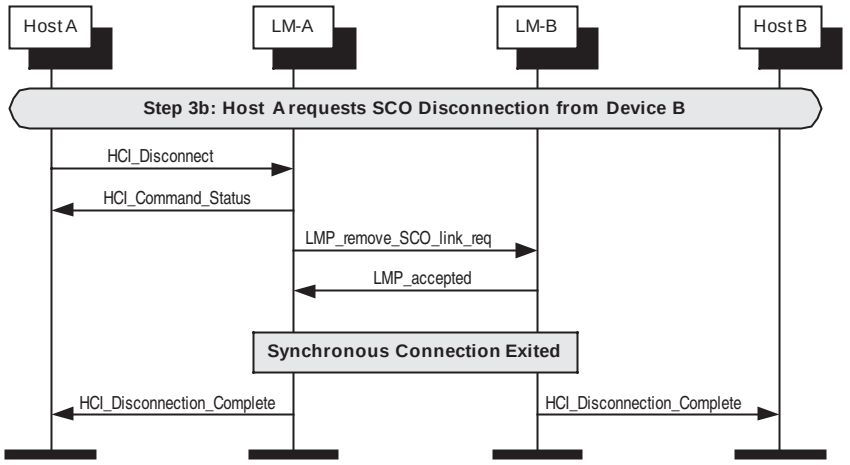


Figure 147—Synchronous disconnection of SCO connection

12.6 SNIFF, HOLD, and PARK

Entry into SNIFF mode, HOLD mode, or PARK state requires an established ACL connection.

12.6.1 SNIFF mode

The HCI_Sniff_Mode command is used to enter SNIFF mode. The HCI_Exit_Sniff_Mode command is used to exit SNIFF mode.

Step 1: Host requests to enter SNIFF mode. Multiple LMP_sniff_req PDUs may be sent as the parameters for SNIFF mode are negotiated (see Figure 148).

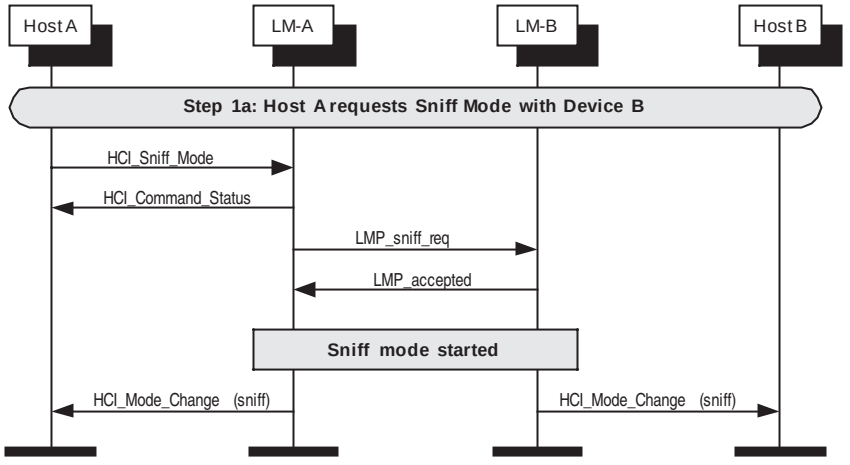


Figure 148—SNIFF mode request

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2: Host requests to exit SNIFF mode (see Figure 149).

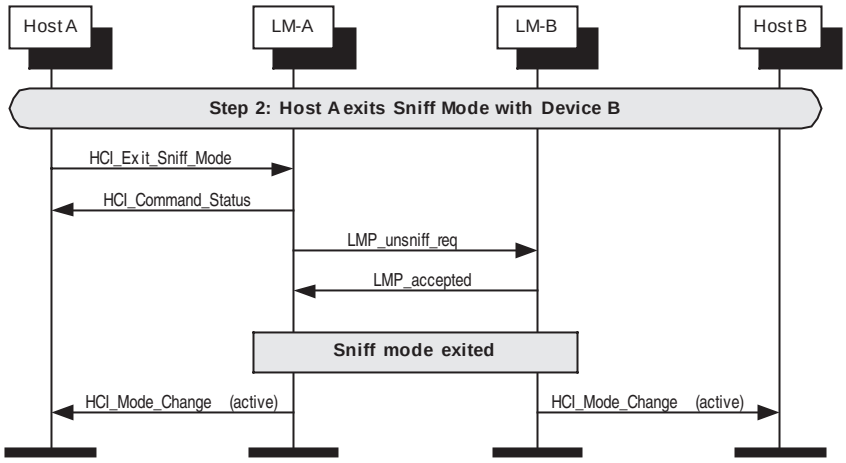


Figure 149—Exit SNIFF mode request

12.6.2 HOLD mode

The HCI_Hold_Mode command can be used to place a device into HOLD mode. The controller may do this by either negotiating the HOLD mode parameters or forcing HOLD mode. HOLD mode will automatically end after the negotiated length of time.

Step 1a: A host requests HOLD mode (see Figure 150).

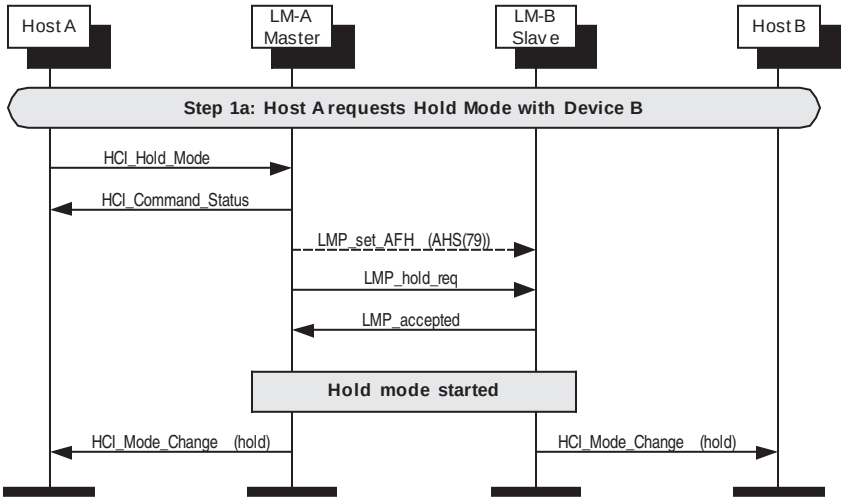


Figure 150—HOLD request

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1b: A host may force HOLD mode (see Figure 151).

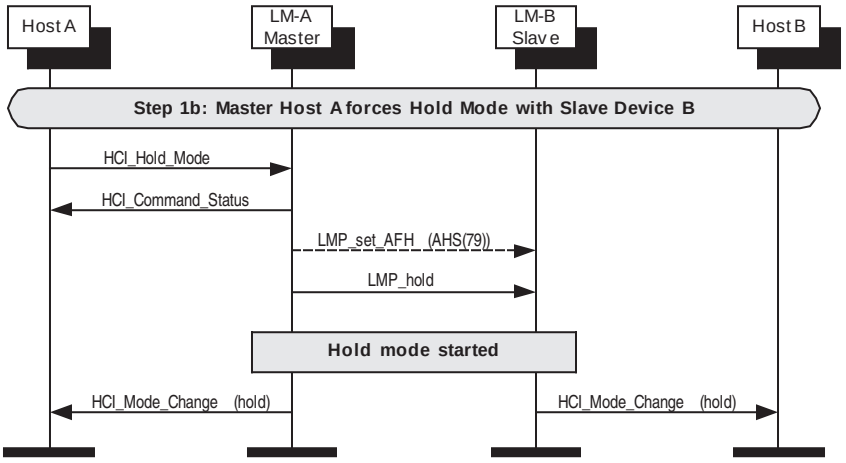


Figure 151—Master forces HOLD mode

Step 1c: A slave device requests HOLD mode (see Figure 152).

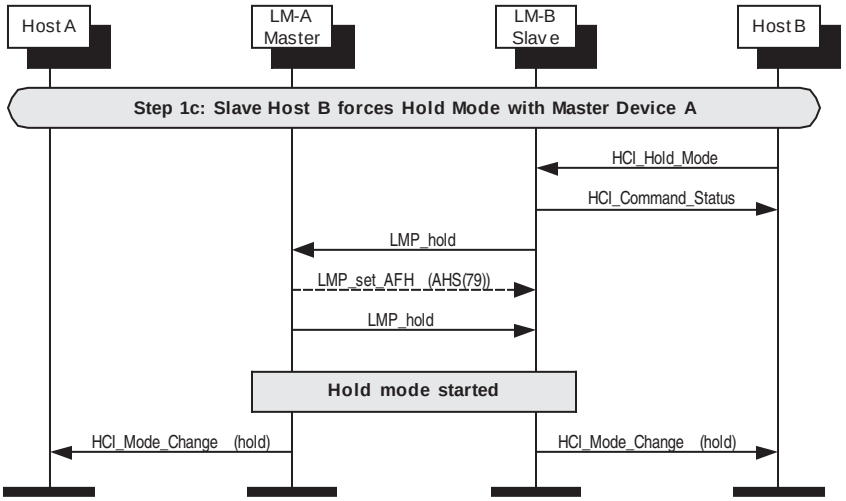


Figure 152—Slave forces HOLD mode

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2: When HOLD mode completes, the hosts are notified using the HCI_Mode_Change event (see Figure 153).

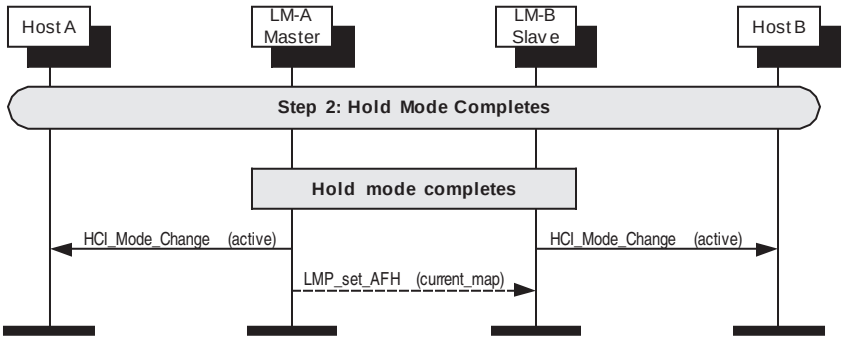


Figure 153—HOLD mode completes

12.6.3 PARK state

PARK state can be entered by using the HCI_Park_State command.

Step 1a: The master requests to place the slave in PARK state. Before sending the LMP_park_req PDU, the master may disable AFH by setting the connection into AHS(79) (see Figure 154).

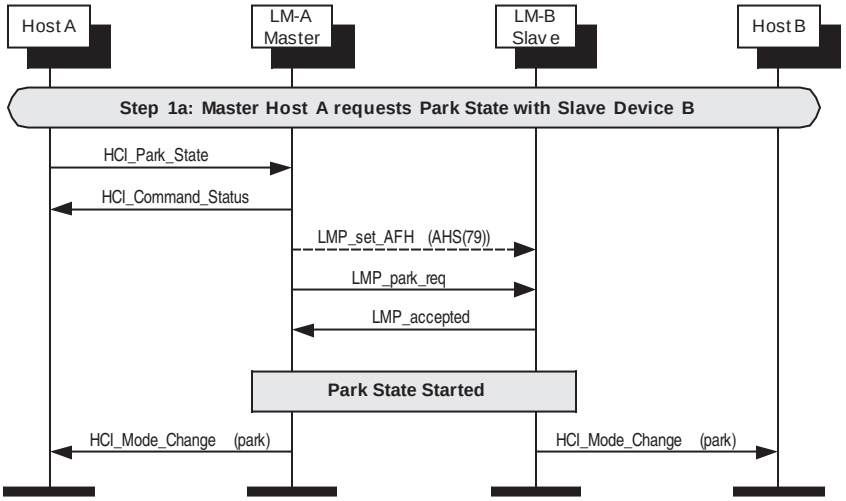


Figure 154—PARK state request from master

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 1b: The slave requests to be placed in PARK state. Before sending the LMP_park_req PDU back to the slave, the master may disable AFH by setting the connection into AHS(79) (see Figure 155).

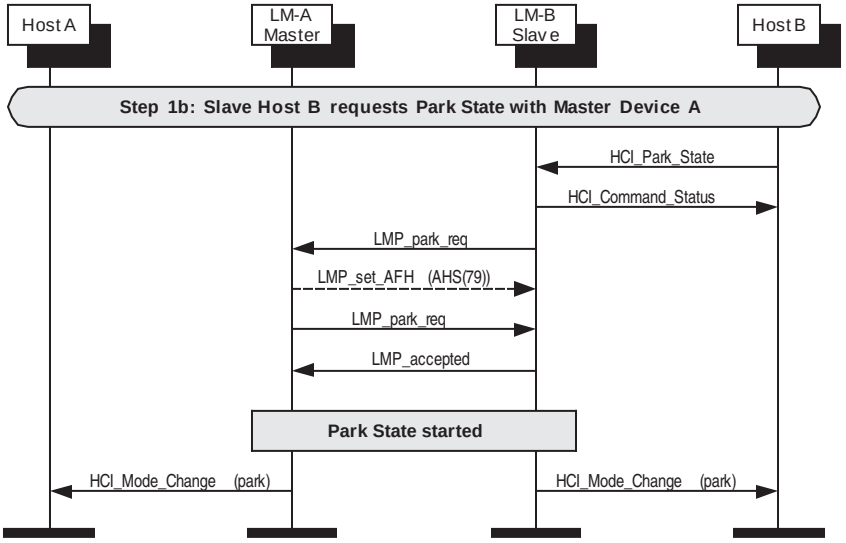


Figure 155—PARK state request from slave

Step 2: When in PARK state, a slave still needs to be unparked for link supervision purposes. The master sends an LMP_unpark_PM_ADDR_req PDU or an LMP_unpark_BD_ADDR_req PDU to the slave during the beacon. Only the PM_ADDR version is illustrated in this MSC (see Figure 156).

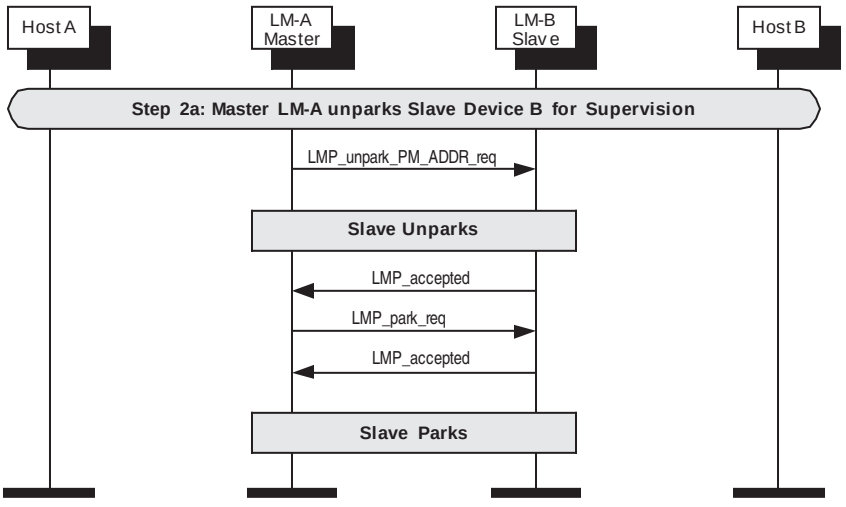


Figure 156—Master unparks slave for supervision

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 3a: A master may unpark a slave to exit PARK state. The master should reenable AFH by setting the current AFH channel map to the unparked slave (see Figure 157).

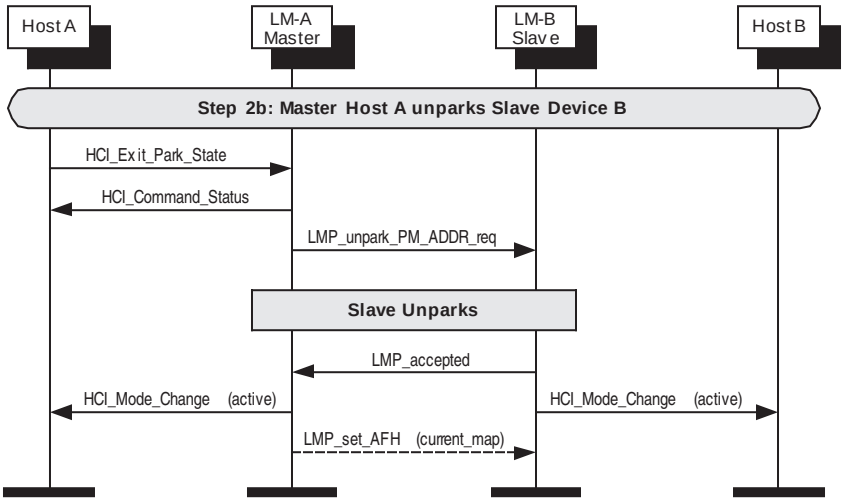


Figure 157—Master exits PARK state with slave

Step 3b: A slave may unpark itself by sending a message in an access window. It will then receive instructions from the master to unpark. The master should reenable AFH by setting the current AFH channel map to the unparked slave (see Figure 158).

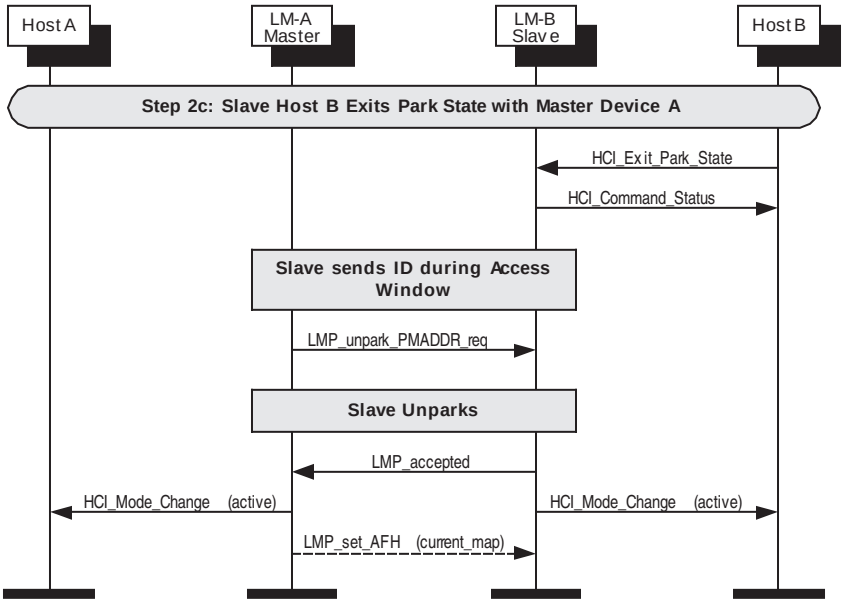


Figure 158—Slave exits PARK state with master

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.7 Buffer management, flow control

Buffer management is very important for resource limited devices. This can be achieved on the HCI using the HCI_Read_Buffer_Size command; the Number Of Completed Packets event; and the HCI_Set_Host_Controller_To_Host_Flow_Control, HCI_Host_Buffer_Size, and HCI_Host_Number_Of_Completed_Packets commands.

Step 1: During initialization, the host reads the buffer sizes available in the controller. When an HCI data packet has been transferred to the remote device and a BB acknowledgment has been received for this data packet, then an HCI_Number_Of_Completed_Packets event will be generated (see Figure 159).

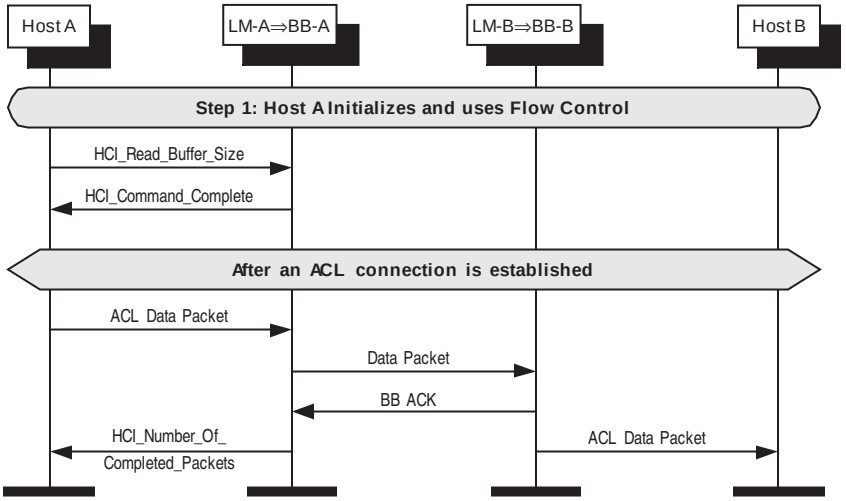


Figure 159—Host-to-controller flow control

Step 2: During initialization, the host notifies the controller that host flow control shall be used and then indicates the host buffer sizes available. When a data packet has been received from a remote device, an HCI data packet is sent to the host from the controller, and the host shall acknowledge its receipt by sending a HCI_Host_Number_Of_Completed_Packets command (see Figure 160).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

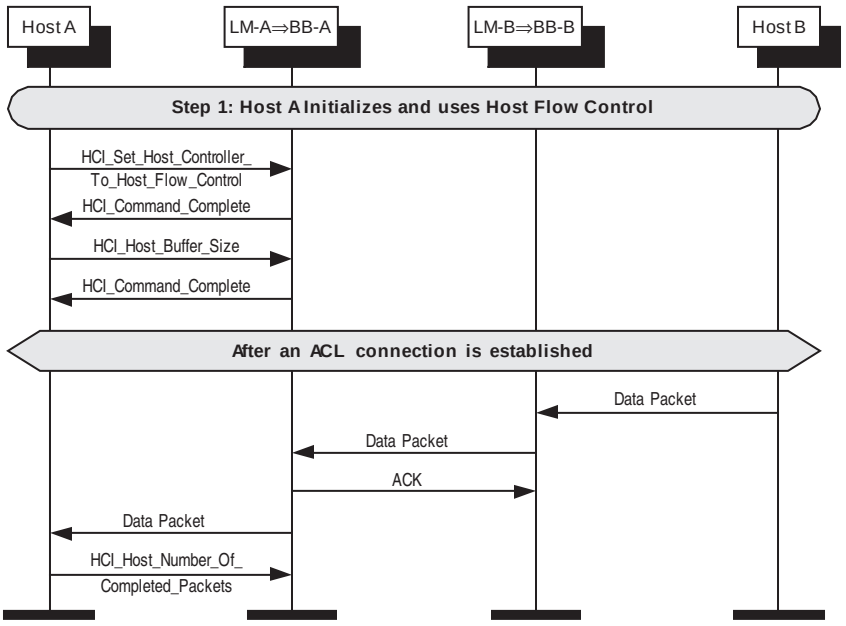


Figure 160—Controller-to-host flow control

12.8 Loopback mode

The loopback modes are used for testing of a device only.

12.8.1 Local loopback mode

The local loopback mode is used to loop back received HCI commands and HCI ACL and HCI synchronous packets sent from the host to the controller.

Step 1: The host enters local loopback mode. Four Connection Complete events are generated and then a Command Complete event (see Figure 161).

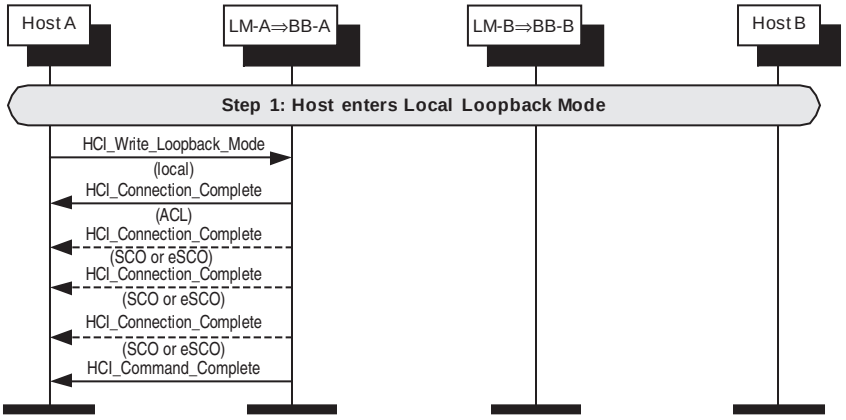


Figure 161—Entering local loopback mode

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 2a: The host sending HCI data packet will receive the exact same data back in HCI data packets from the controller (see Figure 162).

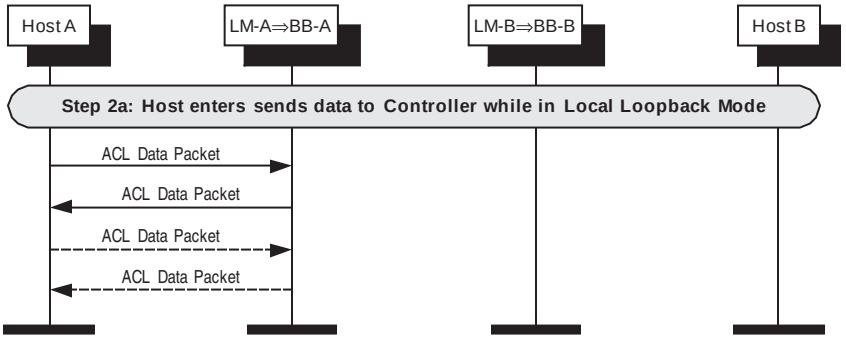


Figure 162—Looping back data in local loopback mode

Step 2b: The host sending most HCI command packets to the controller will receive a Loopback Command event with the contents of the HCI command packet in the payload (see Figure 163).

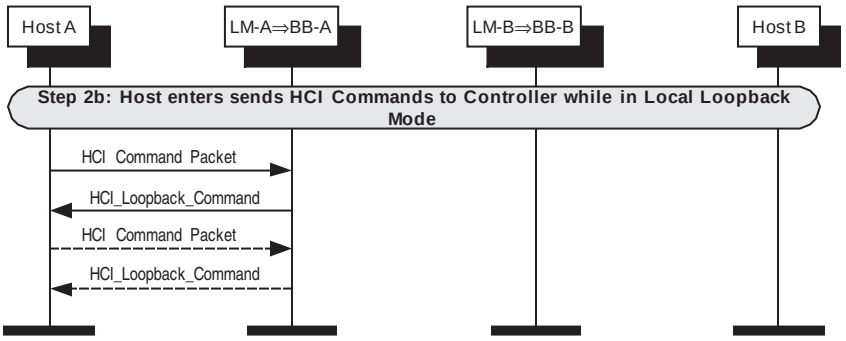


Figure 163—Looping back commands in local loopback mode

Step 3: The host exits local loopback mode. Multiple Disconnection Complete events are generated before the Command Complete event (see Figure 164).

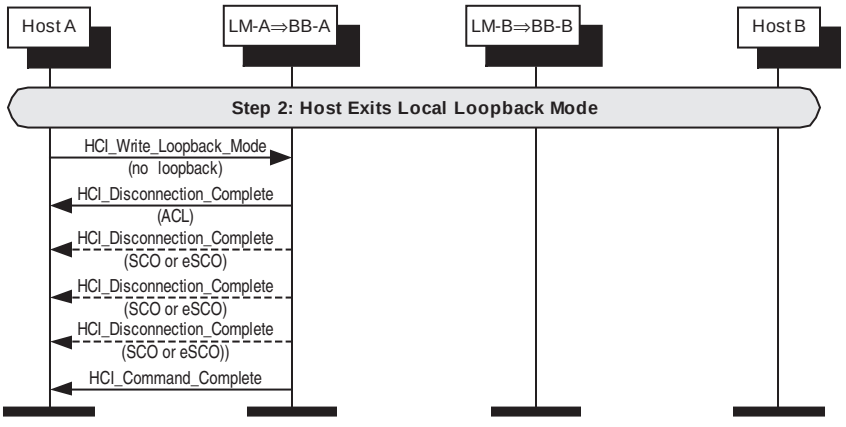


Figure 164—Exiting local loopback mode

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

12.8.2 Remote loopback mode

The remote loopback mode is used to loop back data to a remote device over the air.

Step 1: The remote host first sets up an connection to the local device. The local device then enables remote loopback (see Figure 165).

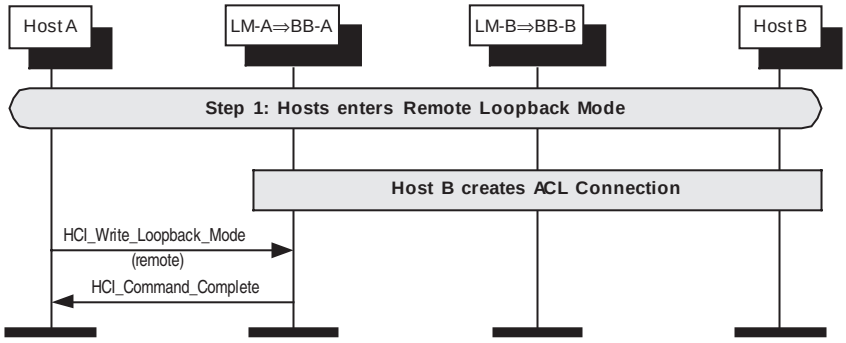


Figure 165—Entering remote loopback mode

Step 2: Any data received from the remote host will be looped back in the controller of the local device (see Figure 166).

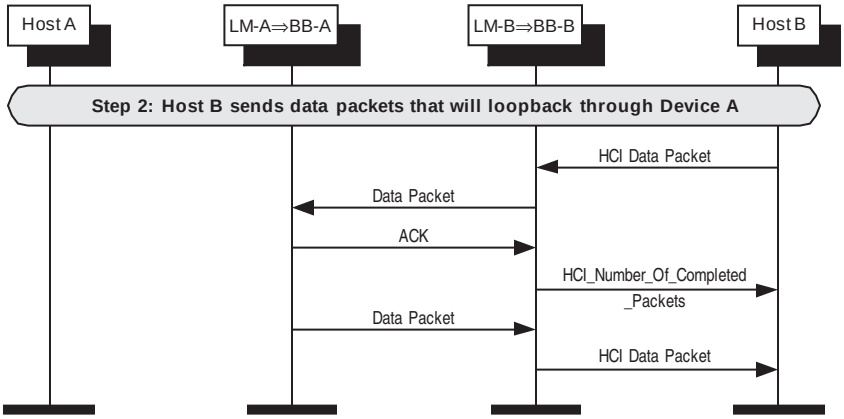


Figure 166—Looping back data in remote loopback mode

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Step 3: The local host exits remote loopback mode. Any connections can then be disconnected by the remote device (see Figure 167).

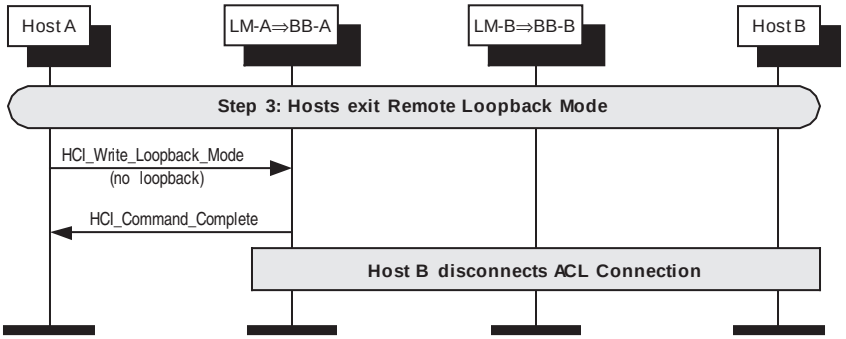


Figure 167—Exiting remote loopback mode

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

13. Security

This clause describes the security system that may be used at the link layer. The encryption, authentication and key generation schemes are specified. The requirements for the supporting process of random number generation are also specified.

13.1 Security overview

The IEEE 802.15.1-2005 wireless technology provides peer-to-peer communications over short distances. In order to provide usage protection and information confidentiality, the system provides security measures both at the application layer and the link layer. These measures are designed to be appropriate for a peer environment. This means that in each device, the authentication and encryption routines are implemented in the same way. Four different entities are used for maintaining security at the link layer: an IEEE 802.15.1-2005 device address (BD_ADDR), two secret keys, and a pseudo-random number that shall be regenerated for each new transaction. The four entities and their sizes are summarized in Table 488.

Table 488—Entities used in authentication and encryption procedures

Entity	Size
BD_ADDR	48 bits
Private user key, authentication	128 bits
Private user key, encryption configurable length (byte-wise)	8–128 bits
Random number	128 bits

The BD_ADDR is the 48-bit address. The BD_ADDR can be obtained via user interactions or, automatically, via an inquiry routine by a device.

The secret keys are derived during initialization and are never disclosed. The encryption key is derived from the authentication key during the authentication process. For the authentication algorithm, the size of the key used is always 128 bits. For the encryption algorithm, the key size may vary between 1 and 16 octets (8–128 bits). The size of the encryption key is configurable for two reasons. The first has to do with the many different requirements imposed on cryptographic algorithms in different countries—both with respect to export regulations and official attitudes toward privacy in general. The second reason is to facilitate a future upgrade path for the security without the need of a costly redesign of the algorithms and encryption hardware; increasing the effective key size is the simplest way to combat increased computing power at the opponent side.

The encryption key is entirely different from the authentication key (even though the latter is used when creating the former, as is described in 13.6.4). Each time encryption is activated, a new encryption key shall be generated. Thus, the lifetime of the encryption key does not necessarily correspond to the lifetime of the authentication key.

It is anticipated that the authentication key will be more static in its nature than the encryption key: once the authentication key is established, the particular application running on the device decides when, or if, to change it. To underline the fundamental importance of the authentication key to a specific link, it is often referred to as the *link key.*

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The random number is a pseudo-random number that can be derived from a random or pseudo-random process in the device. This is not a static parameter; it will change frequently.

In the remainder of this clause, the terms *user* and *application* will be used interchangeably to designate the entity that is at either side.

13.2 Random number generation

Each device has a pseudo-random number generator. Pseudo-random numbers are used for many purposes within the security functions, for instance, for the challenge-response scheme, for generating authentication and encryption keys, etc. Ideally, a true random generator based on some physical process with inherent randomness should be used as a seed. Examples of such processes are thermal noise from a semiconductor or resistor and the frequency instability of a free-running oscillator. For practical reasons, a software-based solution with a pseudo-random generator is probably preferable. In general, it is quite difficult to classify the randomness of a pseudo-random sequence. Within this standard, the requirements placed on the random numbers used are nonrepeating and randomly generated.

The expression *nonrepeating* means that it shall be highly unlikely that the value will repeat itself within the lifetime of the authentication key. For example, a nonrepeating value could be the output of a counter that is unlikely to repeat during the lifetime of the authentication key, e.g., a date/time stamp.

The expression *randomly generated* means that it shall not be possible to predict its value with a chance that is significantly larger than 0 (e.g., greater than $1/2^L$ or a key length of L bits).

The LM may use such a generator for various purposes, i.e., whenever a random number is needed (such as the RANDs, the unit keys, K_{init}, K_{master}, and random back-off or waiting intervals).

13.3 Key management

It is important that the encryption key size within a specific device cannot be set by the user; this should be a factory preset entity. In order to prevent the user from overriding the permitted key size, the BB processing shall not accept an encryption key given from higher software layers. Whenever a new encryption key is required, it shall be created as defined in 13.6.4.

Changing a link key shall also be done through the defined BB procedures. Depending on what kind of link key it is, different approaches are required. The details are found in 13.3.2.7.

13.3.1 Key types

The link key is a 128-bit random number, which is shared between two or more parties and is the base for all security transactions between these parties. The link key itself is used in the authentication routine. Moreover, the link key is used as one of the parameters when the encryption key is derived.

In the following, a *session* is defined as the time interval for which the device is a member of a particular piconet. Thus, the session terminates when the device disconnects from the piconet.

The link keys are either semi-permanent or temporary. A semi-permanent link key may be stored in nonvolatile memory and may be used after the current session is terminated. Consequently, once a semi-permanent link key is defined, it may be used in the authentication of several subsequent connections between the devices sharing it. The designation *semi-permanent* is justified by the possibility of changing it. How to do this is described in 13.3.2.7.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The lifetime of a temporary link key is limited by the lifetime of the current session; it shall not be reused in a later session. Typically, in a point-to-multipoint configuration where the same information is to be distributed securely to several recipients, a common encryption key is useful. To achieve this, a special link key (denoted *master key*) may temporarily replace the current link keys. The details of this procedure are found in 13.3.2.6.

In the following, the *current link key* is the link key in use at the current moment. It can be semi-permanent or temporary. Thus, the current link key is used for all authentications and all generation of encryption keys in the ongoing connection (session).

In order to accommodate different types of applications, four types of link keys have been defined:

— The combination key K_{AB}
— The unit key K_A
— The temporary key K_{master}
— The initialization key K_{init}

NOTE—The use of unit keys is deprecated since it is implicitly insecure.

In addition to these keys, there is an encryption key, denoted K_c. This key is derived from the current link key. Whenever encryption is activated by an LM command, the encryption key shall be changed automatically. The purpose of separating the authentication key and encryption key is to facilitate the use of a shorter encryption key without weakening the strength of the authentication procedure. There are no governmental restrictions on the strength of authentication algorithms. However, in some countries, such restrictions exist on the strength of encryption algorithms.

The combination key K_{AB} and the unit key K_A are functionally indistinguishable; the difference is in the way they are generated. The unit key K_A is generated in, and therefore dependent on, a single device A. The unit key shall be generated once at installation of the device; thereafter, it is very rarely changed. The combination key is derived from information in both devices A and B and is, therefore, always dependent on two devices. The combination key is derived for each new combination of two devices.

It depends on the application or the device whether a unit key or a combination key is used. Devices that have little memory to store keys, or are installed in equipment that will be accessible to a large group of users, should use their own unit key. In that case, they have to store only a single key. Applications that require a higher security level should use the combination keys. These applications will require more memory since a combination key for each link to a different device has to be stored.

The master key K_{master} shall be used only during the current session. It shall replace the original link key only temporarily. For example, this may be utilized when a master wants to reach more than two devices simultaneously using the same encryption key (see 13.3.2.6).

The initialization key K_{init} shall be used as the link key during the initialization process when no combination or unit keys have been defined and exchanged yet or when a link key has been lost. The initialization key protects the transfer of initialization parameters. The key is derived from a random number, an *L*-octet PIN code, and a BD_ADDR. This key shall be used only during initialization.

The PIN may be a fixed number provided with the device. This may occur, for example, when there is no user interface as in a public switched telephone network (PSTN) plug. Alternatively, the PIN can be selected by the user and then entered in both devices that are to be matched. The latter procedure should be used when both devices have a user interface, e.g., a phone and a laptop. Entering a PIN in both devices is more secure than using a fixed PIN in one of the devices and should be used whenever possible. Even if a fixed PIN is used, it shall be possible to change the PIN; this is in order to prevent reinitialization by users who

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

once had possession of the PIN. If no PIN is available, a default value of zero may be used. The length of this default PIN is one byte, PIN(default) = 0x00. This default PIN may be provided by the host.

For many applications, the PIN code will be a relatively short string of numbers. Typically, it may consist of only four decimal digits. Even though this gives sufficient security in many cases, there exist countless other, more sensitive, situations where this is not reliable enough. Therefore, the PIN code may be chosen to be any length from 1 to 16 octets. For the longer lengths, the devices exchanging PIN codes may not use mechanical (i.e., human) interaction, but rather may use software at the application layer. For example, this can be a Diffie-Hellman key agreement, where the exchanged key is passed on to the K_{init} generation process in both devices, just as in the case of a shorter PIN code.

13.3.2 Key generation and initialization

The link keys must be generated and distributed among the devices in order to be used in the authentication procedure. Since the link keys shall be secret, they shall not be obtainable through an inquiry routine in the same way as the device addresses. The exchange of the keys takes place during an initialization phase, which shall be carried out separately for each two devices that are using authentication and encryption. The initialization procedures consist of the following five parts:

— Generation of an initialization key
— Generation of link key
— Link key exchange
— Authentication
— Generation of encryption key in each device (optional)

After the initialization procedure, the devices can proceed to communicate, or the link can be disconnected. If encryption is implemented, the E_0 algorithm shall be used with the proper encryption key derived from the current link key. For any new connection established between devices A and B, they should use the common link key for authentication, instead of once more deriving K_{init} from the PIN. A new encryption key derived from that particular link key shall be created next time encryption is activated.

If no link key is available, the LM shall automatically start an initialization procedure.

13.3.2.1 Generation of initialization key K_{init}

A link key is used temporarily during initialization, the initialization key K_{init}. This key shall be derived by the E_{22} algorithm from a BD_ADDR; a PIN code; the length of the PIN (in octets); and a random number, IN_RAND. The principle is depicted in Figure 182. The 128-bit output from E_{22} shall be used for key exchange during the generation of a link key. When the devices have performed the link key exchange, the initialization key shall be discarded.

When the initialization key is generated, the PIN is augmented with the BD_ADDR. If one device has a fixed PIN, the BD_ADDR of the other device shall be used. If both devices have a variable PIN, the BD_ADDR of the device that received IN_RAND shall be used. If both devices have a fixed PIN, they cannot be paired. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used. This procedure ensures that K_{init} depends on the identity of the device with a variable PIN. A fraudulent device may try to test a large number of PINs by claiming another BD_ADDR each time. It is the application's responsibility to take countermeasures against this threat. If the device address is kept fixed, the waiting interval before the next try may be increased exponentially (see 13.5.1).

The details of the E_{22} algorithm can be found in 13.6.3.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

13.3.2.2 Authentication

The authentication procedure shall be carried out as described in 13.5. During each authentication, a new random number, AU_RAND_A, shall be issued.

Mutual authentication is achieved by first performing the authentication procedure in one direction and then immediately performing the authentication procedure in the opposite direction.

As a side effect of a successful authentication procedure, an auxiliary parameter, the authenticated ciphering offset (ACO), will be computed. The ACO shall be used for ciphering key generation as described in 13.3.2.5.

The claimant/verifier status is determined by the LM.

13.3.2.3 Generation of a unit key

A unit key, K_A, shall be generated when the device is in operation for the first time, i.e., not during each initialization. The unit key shall be generated by the E_{21} algorithm as described in 13.6.3. Once created, the unit key should be stored in nonvolatile memory and very rarely changed. If after initialization the unit key is changed, any previously initialized devices will possess a wrong link key. At initialization, the application must determine which of the two parties will provide the unit key as the link key. Typically, this will be the device with restricted memory capabilities, since this device has to remember only its own unit key. The unit key shall be transferred to the other party and then stored as the link key for that particular party. So, for example, in Figure 168, the unit key of device A, K_A, is being used as the link key for the connection A-B; device A sends the unit key K_A to device B; device B will store K_A as the link key K_{BA}. For another initialization, e.g., with device C, device A will reuse its unit key K_A, whereas device C stores it as K_{CA}.

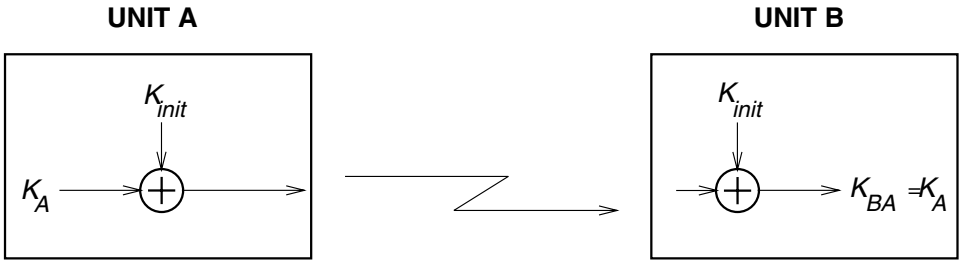


Figure 168—Generating a unit key

When the unit key has been exchanged, the initialization key is discarded in both devices.

13.3.2.4 Generation of a combination key

To use a combination key, it is first generated during the initialization procedure. The combination key is the combination of two numbers generated in devices A and B, respectively. First, each device shall generate a random number, LK_RAND_A and LK_RAND_B. Then, utilizing E_{21} with the random number and their own BD_ADDRs, the two random numbers

$$LK\_K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A) \quad (20)$$

and

$$LK\_K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B), \quad (21)$$

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

shall be created in device A and device B, respectively. These numbers constitute the devices' contribution to the combination key that is to be created. Then, the two random numbers LK_RAND_A and LK_RAND_B shall be exchanged securely by XORing with the current link key K. Thus, device A shall send $K \oplus LK\_RAND_A$ to device B, while device B shall send $K \oplus LK\_RAND_B$ to device A. If this is done during the initialization phase, the link key $K = K_{init}$.

When the random numbers LK_RAND_A and LK_RAND_B have been mutually exchanged, each device shall recalculate the other device's contribution to the combination key. This is possible since each device knows the device address of the other device. Thus, device A shall calculate Equation (21) and device B shall calculate Equation (20). After this, both devices shall combine the two numbers to generate the 128-bit link key. The combining operation is a simple bitwise modulo-2 addition (i.e., XOR). The result shall be stored in device A as the link key K_{AB} and in device B as the link key K_{BA}. When both devices have derived the new combination key, a mutual authentication procedure shall be initiated to confirm the success of the transaction. The old link key shall be discarded after a successful exchange of a new combination key. The message flow between master and slave and the principle for creating the combination key are depicted in Figure 169.

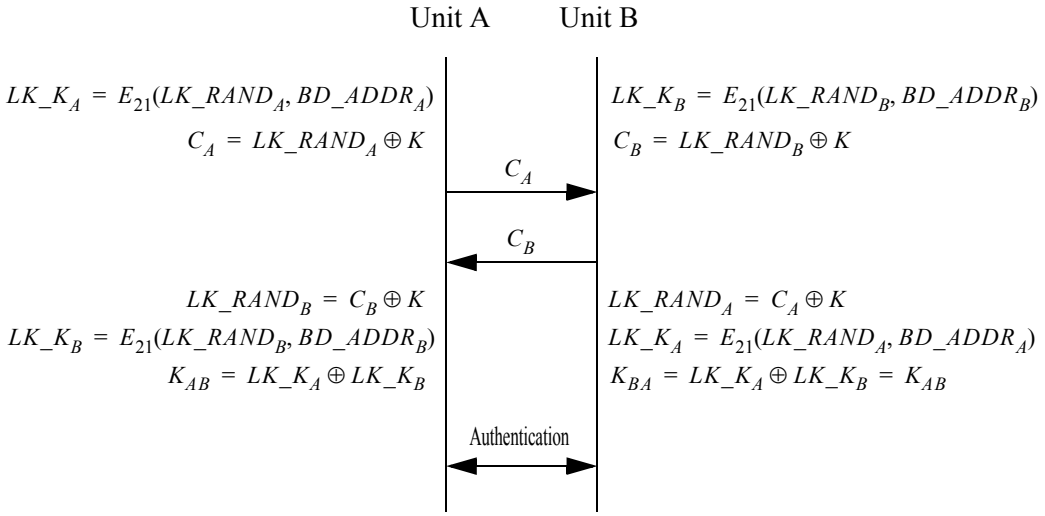


Figure 169—Generating a combination key

The old link key (K) is discarded after the exchange of a new combination key has succeeded.

13.3.2.5 Generating the encryption key

The encryption key K_C is derived by algorithm E_3 from the current link key, a 96-bit ciphering offset (COF) number, and a 128-bit random number. The COF is determined in one of two ways. If the current link key is a master key, then COF shall be derived from the master BD_ADDR. Otherwise, the value of COF shall be set to the value of ACO as computed during the authentication procedure. Therefore,[12]

$$\text{COF} = \begin{cases} \text{BD\_ADDR} \cup \text{BD\_ADDR}, & \text{if link key is a master key} \\ \text{ACO}, & \text{otherwise.} \end{cases} \quad (22)$$

[12] $x \cup y$ denotes the concatenation of x and y.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

There is an explicit call of E_3 when the LM activates encryption. Consequently, the encryption key is automatically changed each time the device enters encryption mode. The details of the key generating function E_3 can be found in 13.6.4.

13.3.2.6 Point-to-multipoint configuration

It is possible for the master to use separate encryption keys for each slave in a point-to-multipoint configuration with ciphering activated. Then, if the application requires more than one slave to listen to the same payload, each slave must be addressed individually. This can cause unwanted capacity loss for the piconet. Moreover, a slave might not be capable of switching between two or more encryption keys in real time (e.g., after looking at the LT_ADDR in the header). Thus, the master cannot use different encryption keys for broadcast messages and individually addressed traffic. Therefore, the master may tell several slave devices to use a common link key (and, hence, indirectly also to use a common encryption key) and may then broadcast the information encrypted. For many applications, this key is only of temporary interest. In the following discussion, this key is denoted by K_{master}.

The transfer of necessary parameters shall be protected by the routine described in 13.3.2.8. After the confirmation of successful reception in each slave, the master shall issue a command to the slaves to replace their respective current link key by the new (temporary) master key. Before encryption can be activated, the master shall also generate and distribute a common EN_RAND to all participating slaves. Using this random number and the newly derived master key, each slave shall generate a new encryption key.

Note that the master must negotiate the encryption key length to use individually with each slave that will use the master key. If the master has already negotiated with some of these slaves, it has knowledge of the sizes that can be accepted. There may be situations where the permitted key lengths of some devices are incompatible. In that case, the master must exclude the limiting device from the group.

When all slaves have received the necessary data, the master can communicate information on the piconet securely using the encryption key derived from the new temporary link key. Each slave in possession of the master key can eavesdrop on all encrypted traffic, not only the traffic intended for itself. The master may tell all participants to fall back to their old link keys simultaneously.

13.3.2.7 Modifying link keys

A link key based on a unit key can be changed. The unit key is created once during first use. Typically, the link key should be changed rather than the unit key, as several devices may share the same unit key as link key (e.g., a printer whose unit key is distributed to all users using the printer's unit key as link key). Changing the unit key will require reinitialization of all devices connecting. Changing the unit key can be justified in some circumstances, e.g., to deny access to all previously allowed devices.

If the key change concerns combination keys, then the procedure is straightforward. The change procedure is identical to the procedure described in Figure 169, using the current value of the combination key as link key. This procedure can be carried out at any time after the authentication and encryption start. Since the combination key corresponds to a single link, it can be modified each time this link is established. This will improve the security of the system since then old keys lose their validity after each session.

Starting up an entirely new initialization procedure is also possible. In that case, user interaction is necessary since a PIN will be required in the authentication and encryption procedures.

13.3.2.8 Generating a master key

The key-change routines described so far are semi-permanent. To create the master link key, which can replace the current link key during a session (see Clause 13.3.2.6), other means are needed. First, the master shall create a new link key from two 128-bit random numbers, RAND1 and RAND2. This shall be done by

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

$$K_{master} = E_{22}(\text{RAND1, RAND2, 16}). \quad (23)$$

This key is a 128-bit random number. The reason for using the output of E_{22}, and not directly choosing a random number as the key, is to avoid possible problems with degraded randomness due to a poor implementation of the random number generator within the device.

Then, a third random number, RAND, shall be transmitted to the slave. Using E_{22} with the current link key and RAND as inputs, both the master and the slave shall compute a 128-bit overlay. The master shall send the bitwise XOR of the overlay and the new link key to the slave. The slave, who knows the overlay, shall recalculate K_{master}. To confirm the success of this transaction, the devices shall perform a mutual authentication procedure using the new link key. This procedure shall then be repeated for each slave that receives the new link key. The ACO values from the authentications shall not replace the current ACO, as this ACO is needed to (re)compute a ciphering key when the master falls back to the previous (nontemporary) link key.

The master activates encryption by an LM command. Before activating encryption, the master shall ensure that all slaves receive the same random number, EN_RAND, since the encryption key is derived through the means of E_3 individually in all participating devices. Each slave shall compute a new encryption key as follows:

$$K_C = E_3(K_{master}, \text{EN\_RAND, COF}) \quad (24)$$

where the value of COF shall be derived from the master's BD_ADDR as specified by Equation (22). The details of the encryption key generating function are described in 13.6.4. The message flow between the master and the slave when generating the master key is depicted in Figure 170. Note that in this case the ACO produced during the authentication is not used when computing the ciphering key.

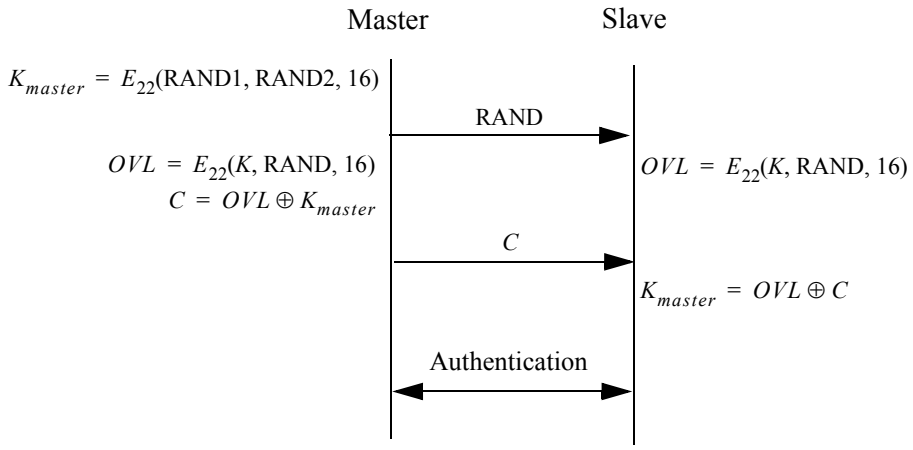


Figure 170—Master link key distribution and computation of the corresponding encryption key

13.4 Encryption

User information can be protected by encryption of the packet payload; the access code and the packet header shall never be encrypted. The encryption of the payload shall be carried out with a stream cipher called E_0 that shall be resynchronized for every payload. The overall principle is shown in Figure 171.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

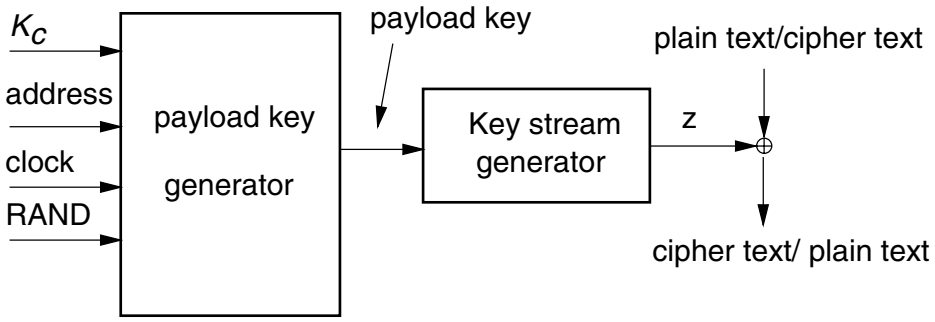


Figure 171—Stream ciphering for IEEE 802.15.1-2005 with E_0

The stream cipher system E_0 shall consist of three parts:

— The first part performs initialization (generation of the payload key). The payload key generator shall combine the input bits in an appropriate order and shall shift them into the four LFSRs used in the key stream generator.
— The second part generates the key stream bits. This shall use a method derived from the summation stream cipher generator attributable to Massey and Rueppel. The second part is the main part of the cipher system, as it will also be used for initialization.
— The third part performs encryption and decryption.

The Massey and Rueppel method has been thoroughly investigated, and there exist good estimates of its strength with respect to presently known methods for cryptanalysis. Although the summation generator has weaknesses that can be used in correlation attacks, the high resynchronization frequency will disrupt such attacks.

13.4.1 Encryption key size negotiation

Each device implementing the BB shall have a parameter defining the maximal allowed key length, L_{max}, $1 \le L_{max} \le 16$ (number of octets in the key). For each application using encryption, a number L_{min} shall be defined indicating the smallest acceptable key size for that particular application. Before generating the encryption key, the devices involved shall negotiate to decide the key size to use.

The master shall send a suggested value, $L_{sug}^{(M)}$, to the slave. Initially, the suggested value shall be set to $L_{max}^{(M)}$. If $L_{min}^{(S)} \le L_{sug}^{(M)}$ and the slave supports the suggested length, the slave shall acknowledge; and this value shall be the length of the encryption key for this link. However, if both conditions are not fulfilled, the slave shall send a new proposal, $L_{sug}^{(S)} < L_{sug}^{(M)}$, to the master. This value shall be the largest among all supported lengths less than the previous master suggestion. Then, the master shall perform the corresponding test on the slave suggestion. This procedure shall be repeated until a key length agreement is reached, or one device aborts the negotiation. An abort may be caused by lack of support for L_{sug} and all smaller key lengths or if $L_{sug} < L_{min}$ in one of the devices. In the case of an abort, link encryption cannot be employed.

The possibility of a failure in setting up a secure link is an unavoidable consequence of letting the application decide whether to accept or reject a suggested key size. However, this is a necessary precaution. Otherwise, a fraudulent device could enforce a weak protection on a link by claiming a small maximum key size.

13.4.2 Encryption of broadcast messages

There may be three settings for the BB regarding encryption:

— *No encryption.* This is the default setting. No messages are encrypted.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— *Point-to-point only encryption.* Broadcast messages are not encrypted. This may be enabled either during the connection establishment procedure or after the connection has been established.
— *Point-to-point and broadcast encryption*. All messages are encrypted. This may be enabled after the connection has been established only. This setting should not be enabled unless all affected links share the same master link key as well as the same EN_RAND value, both used in generating the encryption key.

13.4.3 Encryption concept

The possible encryption modes for a slave that possesses a master key are shown in Table 489.

Table 489—Possible encryption modes for a slave in possession of a master key

Broadcast traffic	Individually addressed traffic
No encryption	No encryption
No encryption	Encryption, K_{master}
Encryption, K_{master}	Encryption, K_{master}

For the encryption routine, a stream cipher algorithm is used in which ciphering bits are bitwise modulo-2 added to the data stream to be sent over the air interface. The payload is ciphered after the CRC bits are appended, but prior to the FEC encoding.

Each packet payload shall be ciphered separately. The cipher algorithm E_0 uses the master device address (BD_ADDR), 26 bits of the master real-time clock ($CLK_{26–1}$), and the encryption key K_C as input (see Figure 172) (where it is assumed that device A is the master).

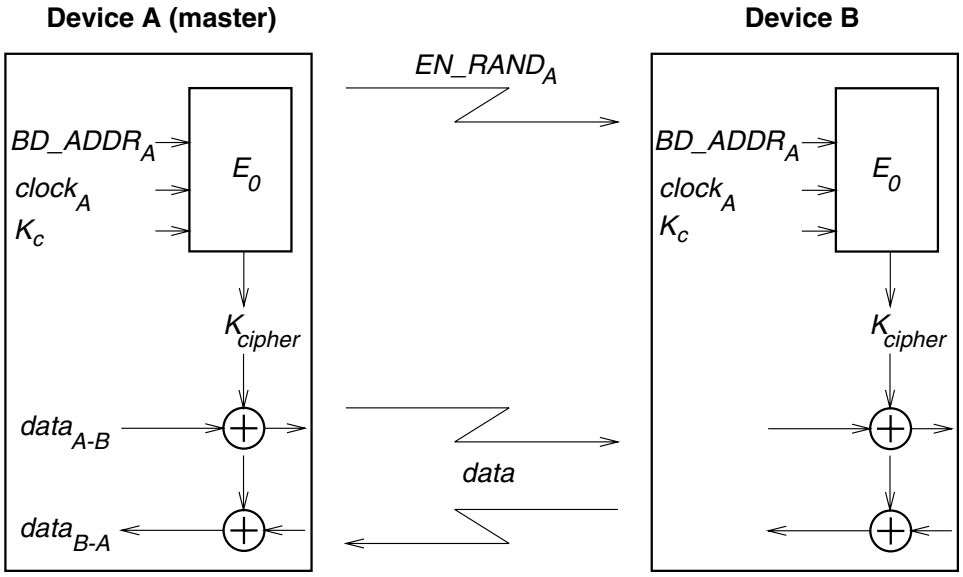


Figure 172—Functional description of the encryption procedure

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The encryption key K_C is derived from the current link key, COF, and a random number, EN_RAND_A (see 13.6.4). The random number shall be issued by the master before entering encryption mode. Note that EN_RAND_A is publicly known since it is transmitted as plain text over the air.

Within the E_0 algorithm, the encryption key K_C is modified into another key denoted K'_C. The maximum effective size of this key shall be factory preset and may be set to any multiple of eight between one and sixteen (8–128 bits). The procedure for deriving the key is described in 13.4.5.

The real-time clock is incremented for each slot. The E_0 algorithm shall be reinitialized at the start of each new packet (i.e., for master-to-slave as well as for slave-to-master transmission). By using $CLK_{26–1}$ at least 1 bit is changed between two transmissions. Thus, a new keystream is generated after each reinitialization. For packets covering more than a single slot, CLK as found in the first slot shall be used for the entire packet.

The encryption algorithm E_0 generates a binary keystream, K_{cipher}, which shall be modulo-2 added to the data to be encrypted. The cipher is symmetric; decryption shall be performed in exactly the same way using the same key as used for encryption.

13.4.4 Encryption algorithm

The system uses LFSRs whose output is combined by a simple finite state machine (called the *summation combiner*) with 16 states. The output of this state machine is the key stream sequence or, during initialization phase, the randomized initial start value. The algorithm uses an encryption key (K_C), a 48-bit address, the master clock bits $CLK_{26–1}$, and a 128-bit RAND value. Figure 173 shows the setup.

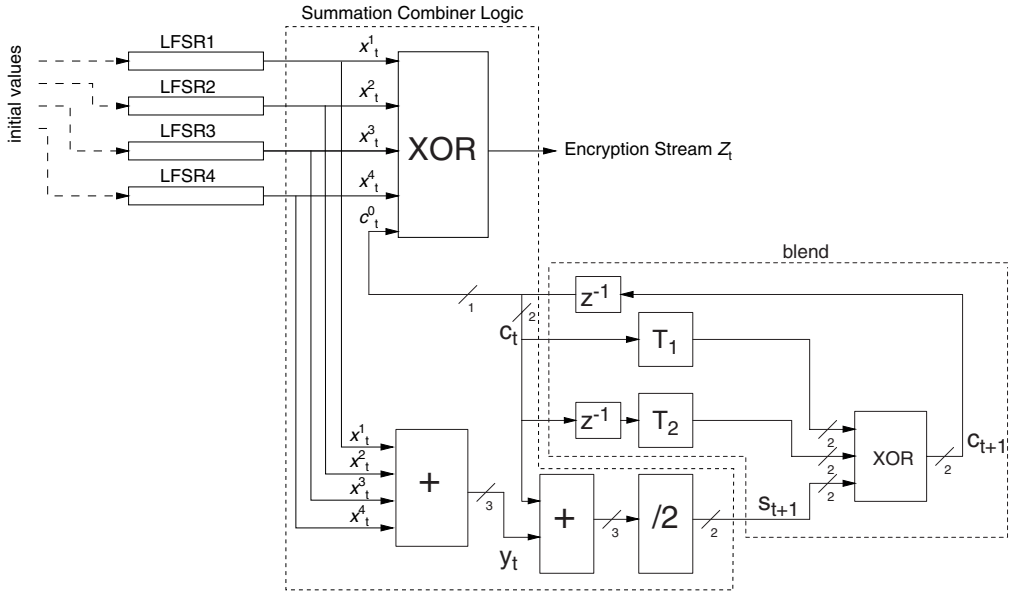


Figure 173—Concept of the encryption engine

There are four LFSRs ($LFSR_1$,...,$LFSR_4$) of lengths $L_1 = 25$, $L_2 = 31$, $L_3 = 33$, and, $L_4 = 39$, with feedback polynomials as specified in Table 490. The total length of the registers is 128. These polynomials are all primitive. The Hamming weight of all the feedback polynomials is chosen to be five—a reasonable trade-off between reducing the number of required XOR gates in the hardware implementation and obtaining good statistical properties of the generated sequences.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 490—The four primitive feedback polynomials

i	L_i	Feedback $f_i(t)$	Weight
1	25	$t^{25} + t^{20} + t^{12} + t^8 + 1$	5
2	31	$t^{31} + t^{24} + t^{16} + t^{12} + 1$	5
3	33	$t^{33} + t^{28} + t^{24} + t^4 + 1$	5
4	39	$t^{39} + t^{36} + t^{28} + t^4 + 1$	5

Let x_t^i denote the tth symbol of LFSR_i. The value y_t is derived from the four-tuple $x_t^1, \ldots, x_t^4$ using the Equation (25):

$$y_t = \sum_{i=1}^{4} x_t^i, \tag{25}$$

where the sum is over the integers. Thus y_t can take the values 0, 1, 2, 3, or 4. The output of the summation generator is obtained by Equation (26), Equation (27), and Equation (28):

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\}, \tag{26}$$

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\}, \tag{27}$$

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \tag{28}$$

where $T_1[.]$ and $T_2[.]$ are two different linear bijections over GF(4). Suppose GF(4) is generated by the irreducible polynomial $x^2 + x + 1$, and let α be a zero of this polynomial in GF(4). The mappings T_1 and T_2 are now defined as follows:

$$T_1: \text{GF}(4) \rightarrow \text{GF}(4)$$

$$x \mapsto x$$

$$T_2: \text{GF}(4) \rightarrow \text{GF}(4)$$

$$x \mapsto (\alpha + 1)x.$$

The elements of GF(4) can be written as binary vectors. This is summarized in Table 491.

Table 491—The mappings $\mathbf{T_1}$ and $\mathbf{T_2}$

x	$T_1[x]$	$T_2[x]$
00	00	00
01	01	11
10	10	01
11	11	10

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Since the mappings are linear, they can be implemented using XOR gates, i.e.,

T_1: $(x_1, x_0) \mapsto (x_1, x_0)$,

T_2: $(x_1, x_0) \mapsto (x_0, x_1 \oplus x_0)$.

13.4.4.1 The operation of the cipher

Figure 174 gives an overview of the operation in time. The encryption algorithm shall run through the initialization phase before the start of transmission or reception of a new packet. Thus, for multislot packets, the cipher is initialized using the clock value of the first slot in the multislot sequence.

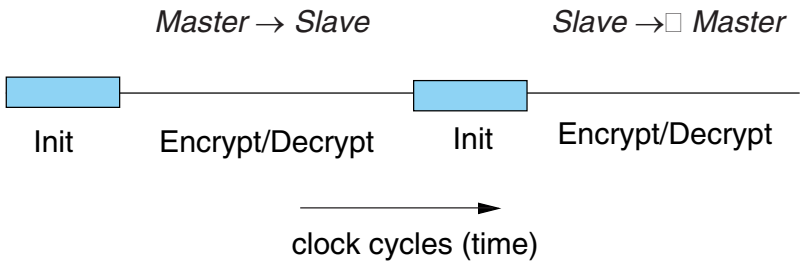


Figure 174—Overview of the operation of the encryption engine

Between each start of a packet (TX or RX), the LFSRs are reinitialized.

13.4.5 LFSR initialization

The key stream generator is loaded with an initial value for the four LFSRs (in total, 128 bits) and the 4 bits that specify the values of c_0 and c_{-1}. The 132-bit initial value is derived from four inputs by using the key stream generator. The input parameters are the key K_C, a 128-bit random number (RAND), a 48-bit device address, and the 26 master clock bits CLK_{26-1}.

The effective length of the encryption key may vary between 8 and 128 bits. Note that the actual key length as obtained from E_3 is 128 bits. Then, within E_0, the key length may be reduced by a modulo operation between K_C and a polynomial of desired degree. After reduction, the result is encoded with a block code in order to distribute the starting states more uniformly. The operation shall be as defined in Equation (29).

When the encryption key has been created, the LFSRs are loaded with their initial values. Then, 200 stream cipher bits are created by operating the generator. Of these bits, the last 128 are fed back into the key stream generator as an initial value of the four LFSRs. The values of c_t and c_{t-1} are kept. From this point on, when clocked the generator produces the encryption (decryption) sequence that is bitwise XORed to the transmitted (received) payload data.

In the following, octet *i* of a binary sequence X is $X[i]$. Bit 0 of X is the LSB. Then, the LSB of $X[i]$ corresponds to bit $8i$ of the sequence X, the MSB of $X[i]$ is bit $8i + 7$ of X. For instance, bit 24 of the device address is the LSB of BD_ADDR[3].

The details of the initialization shall be as follows:

a) Create the encryption key to use from the 128-bit secret key K_C and the 128-bit publicly known EN_RAND. Let L, $1 \le L \le 16$, be the effective key length in number of octets. The resulting encryption key is K'_C:

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

$$K'_C(x) = g_2^{(L)}(x)(K_C(x) \bmod g_1^{(L)}(x)), \quad (29)$$

where $\deg(g_1^{(L)}(x)) = 8L$ and $\deg(g_2^{(L)}(x)) \le 128 - 8L$. The polynomials are defined in Table 492.

b) Shift the three inputs K'_C, the device address, the clock, and the 6-bit constant 111001 into the LFSRs. In total, 208 bits are shifted in:
   1) Open all switches shown in Figure 175;
   2) Arrange inputs bits as shown in Figure 175; Set the content of all shift register elements to zero. Set $t = 0$.
   3) Start shifting bits into the LFSRs. The rightmost bit at each level of Figure 175 is the first bit to enter the corresponding LFSR.
   4) When the first input bit at level i reaches the rightmost position of $LFSR_i$, close the switch of this LFSR.
   5) At $t = 39$ (when the switch of $LFSR_4$ is closed), reset both blend registers $c_{39} = c_{39-1} = 0$. Up to this point, the content of c_t and c_{t-1} has been of no concern. However, their content will now be used in computing the output sequence.
   6) From now on, output symbols are generated. The remaining input bits are continuously shifted into their corresponding shift registers. When the last bit has been shifted in, the shift register is clocked with input = 0;

   NOTE—When finished, $LFSR_1$ has effectively clocked 30 times with feedback closed, $LFSR_2$ has clocked 24 times, $LFSR_3$ has clocked 22 times, and $LFSR_4$ has effectively clocked 16 times with feedback closed.

c) To mix initial data, continue to clock until 200 symbols have been produced with all switches closed ($t = 239$);

d) Keep blend registers c_t and c_{t-1}; make a parallel load of the last 128 generated bits into the LFSRs according to Figure 176 at $t = 240$;

After the parallel load in item 4, the blend register contents shall be updated for each subsequent clock.

Table 492—Polynomials used when creating K'_c[a]

L	Deg	$g_1^{(L)}$	Deg	$g_2^{(L)}$
1	[8]	00000000 00000000 00000000 0000011d	[119]	00e275a0 abd218d4 cf928b9b bf6cb08f
2	[16]	00000000 00000000 00000000 0001003f	[112]	0001e3f6 3d7659b3 7f18c258 cff6efef
3	[24]	00000000 00000000 00000000 010000db	[104]	000001be f66c6c3a b1030a5a 1919808b
4	[32]	00000000 00000000 00000001 000000af	[96]	00000001 6ab89969 de17467f d3736ad9
5	[40]	00000000 00000000 00000100 00000039	[88]	00000000 01630632 91da50ec 55715247
6	[48]	00000000 00000000 00010000 00000291	[77]	00000000 00002c93 52aa6cc0 54468311
7	[56]	00000000 00000000 01000000 00000095	[71]	00000000 000000b3 f7fffce2 79f3a073
8	[64]	00000000 00000001 00000000 0000001b	[63]	00000000 00000000 a1ab815b c7ec8025
9	[72]	00000000 00000100 00000000 00000609	[49]	00000000 00000000 0002c980 11d8b04d

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 492—Polynomials used when creating K'$_c$[a] *(continued)*

L	Deg	$g_1^{(L)}$	Deg	$g_2^{(L)}$
10	[80]	00000000 00010000 00000000 00000215	[42]	00000000 00000000 0000058e 24f9a4bb
11	[88]	00000000 01000000 00000000 0000013b	[35]	00000000 00000000 0000000c a76024d7
12	[96]	00000001 00000000 00000000 000000dd	[28]	00000000 00000000 00000000 1c9c26b9
13	[104]	00000100 00000000 00000000 0000049d	[21]	00000000 00000000 00000000 0026d9e3
14	[112]	00010000 00000000 00000000 0000014f	[14]	00000000 00000000 00000000 00004377
15	[120]	01000000 00000000 00000000 000000e7	[7]	00000000 00000000 00000000 00000089
16	[128]	1 00000000 00000000 00000000 00000000	[0]	00000000 00000000 00000000 00000001

[a]All polynomials are in hexadecimal notation. The LSB is in the rightmost position.

In Figure 175, all bits are shifted into the LFSRs, starting with the LSB. For instance, from the third octet of the address BD_ADDR[2], first BD_ADDR$_{16}$ is entered, followed by BD_ADDR$_{17}$, etc. Furthermore, CL$_0$ corresponds to CLK$_1$,..., CL$_{25}$ corresponds to CLK$_{26}$.

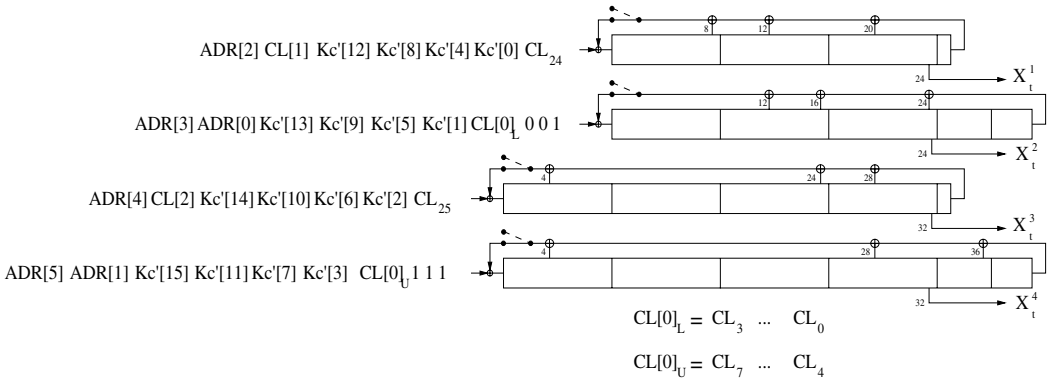


Figure 175—Arranging the input to the LFSRs

Note that the output symbols x_t^i, $i = 1, \ldots, 4$ are taken from the positions 24, 24, 32, and 32 for LFSR$_1$, LFSR$_2$, LFSR$_3$, and LFSR$_4$, respectively (counting the leftmost position as number 1).

In Figure 176, the 128 binary output symbols Z_0,..., Z_{127} are arranged in octets denoted Z[0],..., Z[15]. The LSB of Z[0] corresponds to the first of these symbols; the MSB of Z[15] is the last output from the generator. These bits shall be loaded into the LFSRs according to the figure. It is a parallel load, and no update of the blend registers is done. The first output symbol is generated at the same time. The octets shall be written into the registers with the LSB in the leftmost position (i.e., the opposite of before). For example, Z_{24} is loaded into position 1 of LFSR$_4$.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

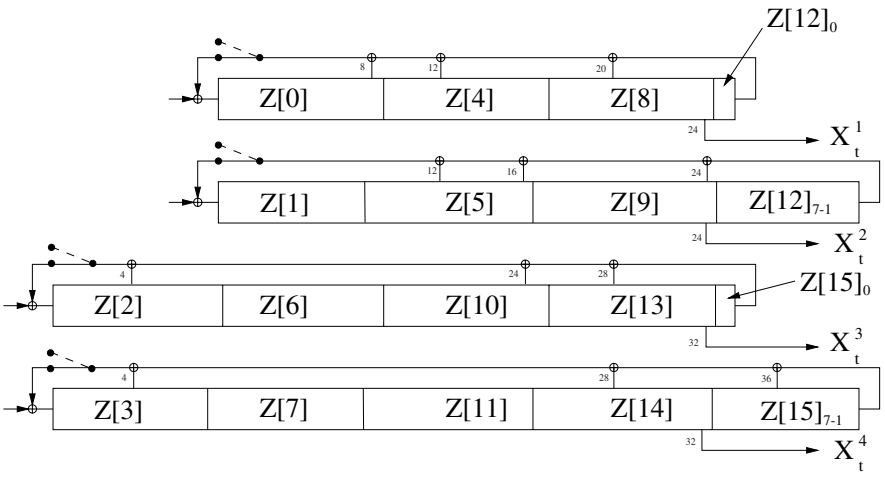


Figure 176—Distribution of the 128 last generated output symbols within the LFSRs

13.4.6 Key stream sequence

When the initialization is finished, the output from the summation combiner is used for encryption/ decryption. The first bit to use shall be the one produced at the parallel load, i.e., at $t = 240$. The circuit shall be run for the entire length of the current payload. Then, before the reverse direction is started, the entire initialization process shall be repeated with updated values on the input parameters.

Sample data of the encryption output sequence can be found in the Bluetooth specification. A necessary, but not sufficient, condition for all compliant implementations of encryption is to produce these encryption streams for identical initialization values.

13.5 Authentication

Authentication uses a challenge-response scheme in which a claimant's knowledge of a secret key is checked through a two-move protocol using symmetric secret keys. The latter implies that a correct claimant/verifier pair share the same secret key, e.g., K. In the challenge-response scheme, the verifier challenges the claimant to authenticate a random input (the challenge), denoted by AU_RAND_A, with an authentication code, denoted by E_1, and return the result signed response (SRES) to the verifier (see Figure 177). This figure also shows that the input to E_1 consists of the tuple AU_RAND_A and the device address (BD_ADDR) of the claimant. The use of this address prevents a simple reflection attack.[13] The secret K shared by devices A and B is the current link key.

The challenge-response scheme for symmetric keys is depicted in Figure 178.

The verifier is not required to be the master. The application indicates which device has to be authenticated. Some applications require only a one-way authentication. However, some peer-to-peer communications should use a mutual authentication in which each device is subsequently the challenger (verifier) in two authentication procedures. The LM shall process authentication preferences from the application to determine in which direction(s) the authentication(s) takes place. For mutual authentication with the devices of Figure 177, after device A has successfully authenticated device B, device B could authenticate device A by sending an AU_RAND_B (different from the AU_RAND_A that device A issued) to device A and deriving the SRES and SRES' from the new AU_RAND_B, the address of device A, and the link key K_{AB}.

If an authentication is successful, the value of ACO as produced by E_1 shall be retained.

[13]The reflection attack actually forms no threat because all service requests are dealt with on a first-in–first-out (FIFO) basis. When preemption is introduced, this attack is potentially dangerous.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

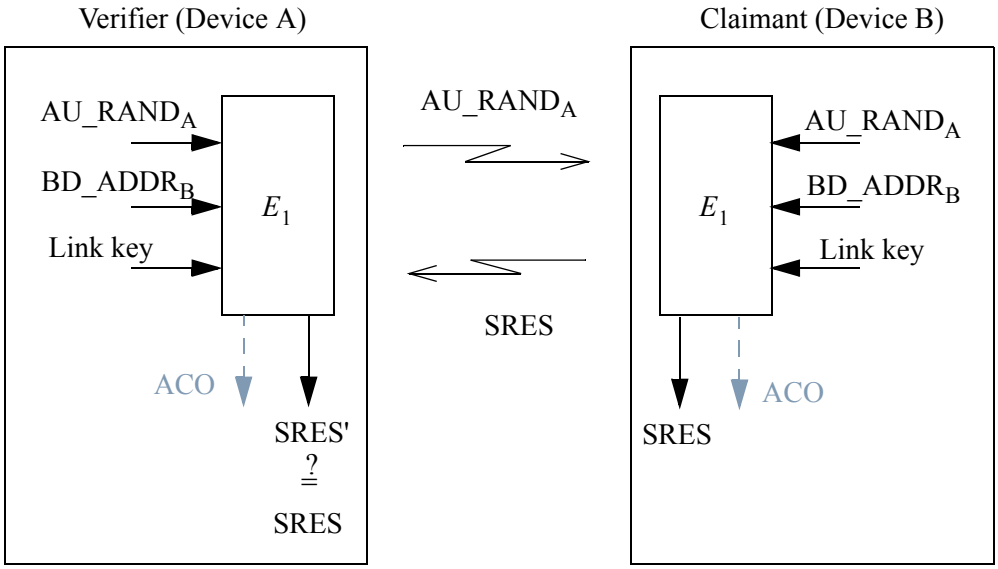


Figure 177—Challenge-response

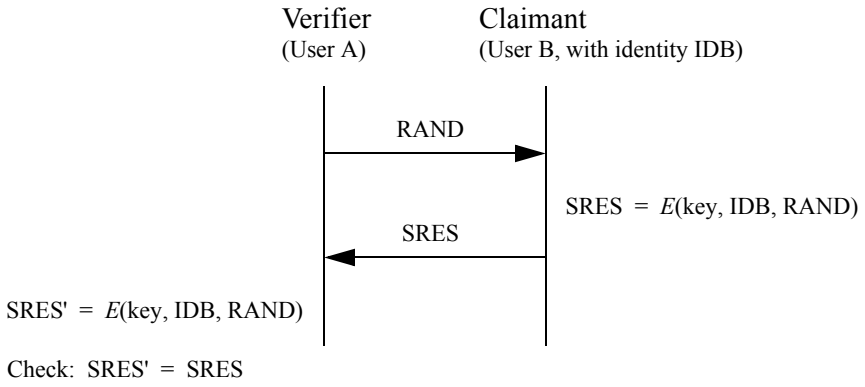


Figure 178—Challenge-response for symmetric key systems

13.5.1 Repeated attempts

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure with the same device address, the waiting interval shall be increased exponentially. In other words, after each failure, the waiting interval before a new attempt can be made could be, for example, twice as long as the waiting interval prior to the previous attempt.[14] The waiting interval shall be limited to a maximum. The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To make the system less vulnerable to denial-of-service attacks, the devices should keep a list of individual waiting intervals for each device with which it has established contact. The size of this list may be restricted to contain only the *N* devices with which the most recent contacts have been made. The number *N* may vary for different devices depending on available memory size and user environment.

[14]Another appropriate value larger than 1 may be used.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

13.6 The authentication and key-generating functions

This subclause describes the algorithms used for authentication and key generation.

13.6.1 The authentication function E_1

The authentication function E_1 is a computationally secure authentication code. E_1 uses the encryption function SAFER+. The algorithm is an enhanced version of an existing 64-bit block cipher SAFER-SK128, and it is freely available. In the following discussion, the block cipher will be denoted as the function A_r, which maps using a 128-bit key, a 128-bit input to a 128-bit output, i.e.,

$$\begin{aligned} A_r\colon\ & \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128} \\ & (k \times x) \mapsto t. \end{aligned} \tag{30}$$

The details of A_r are given in 13.6.2. The function E_1 is constructed using A_r as follows:

$$\begin{aligned} E_1\colon\ & \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32} \times \{0,1\}^{96} \\ & (K, \text{RAND}, \text{address}) \mapsto (\text{SRES}, \text{ACO}), \end{aligned} \tag{31}$$

where $\text{SRES} = Hash(K, \text{RAND}, \text{address}, 6)[0, \ldots, 3]$, where $Hash$ is a keyed hash function defined as[15]

$$\begin{aligned} & Hash\colon \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{128} \\ & (K, I_1, I_2, L) \mapsto A'_r([\tilde{K}], [E(I_2, L) +_{16} (A_r(K, I_1) \oplus_{16} I_1)]), \end{aligned} \tag{32}$$

and where

$$\begin{aligned} & E\colon \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{8 \times 16} \\ & (X[0, \ldots, L-1], L) \mapsto (X[i(\text{mod L})] \ \text{ for } i = 0 \ldots 15), \end{aligned} \tag{33}$$

is an expansion of the L octet word X into a 128-bit word. The function A_r is evaluated twice for each evaluation of E_1. The key K for the second use of A_r (actually A'_r) is offset from K as follows:[16]

$$\begin{aligned} \tilde{K}[0] &= (K[0] + 233) \mod 256, & \tilde{K}[1] &= K[1] \oplus 229, \\ \tilde{K}[2] &= (K[2] + 223) \mod 256, & \tilde{K}[3] &= K[3] \oplus 193, \\ \tilde{K}[4] &= (K[4] + 179) \mod 256, & \tilde{K}[5] &= K[5] \oplus 167, \\ \tilde{K}[6] &= (K[6] + 149) \mod 256, & \tilde{K}[7] &= K[7] \oplus 131, \\ \tilde{K}\{8\} &= K[8] \oplus 233, & \tilde{K}[9] &= (K[9] + 229) \mod 256, \\ \tilde{K}[10] &= K[10] \oplus 223, & \tilde{K}[11] &= (K[11] + 193) \mod 256, \\ \tilde{K}[12] &= K[12] \oplus 179, & \tilde{K}[13] &= (K[13] + 167) \mod 256, \\ \tilde{K}[14] &= K[14] \oplus 149, & \tilde{K}[15] &= (K[15] + 131) \mod 256. \end{aligned} \tag{34}$$

[15]The operator $+_{16}$ denotes bytewise addition mod-256 of the 16 octets, and the operator $\oplus_{16}$ denotes bytewise XORing of the 16 octets.

[16]The constants are the largest primes below 257 for which 10 is a primitive root.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A data flowchart of the computation of E_1 is shown in Figure 179. E_1 is also used to deliver the parameter ACO that is used in the generation of the ciphering key by E_3 [see Equation (22) and Equation (42)]. The value of ACO is formed by the octets 4 through 15 of the output of the hash function defined in Equation (32), i.e.,

$$\text{ACO} = Hash(K, \text{RAND}, \text{address}, 6)[4, \ldots, 15]. \quad (35)$$

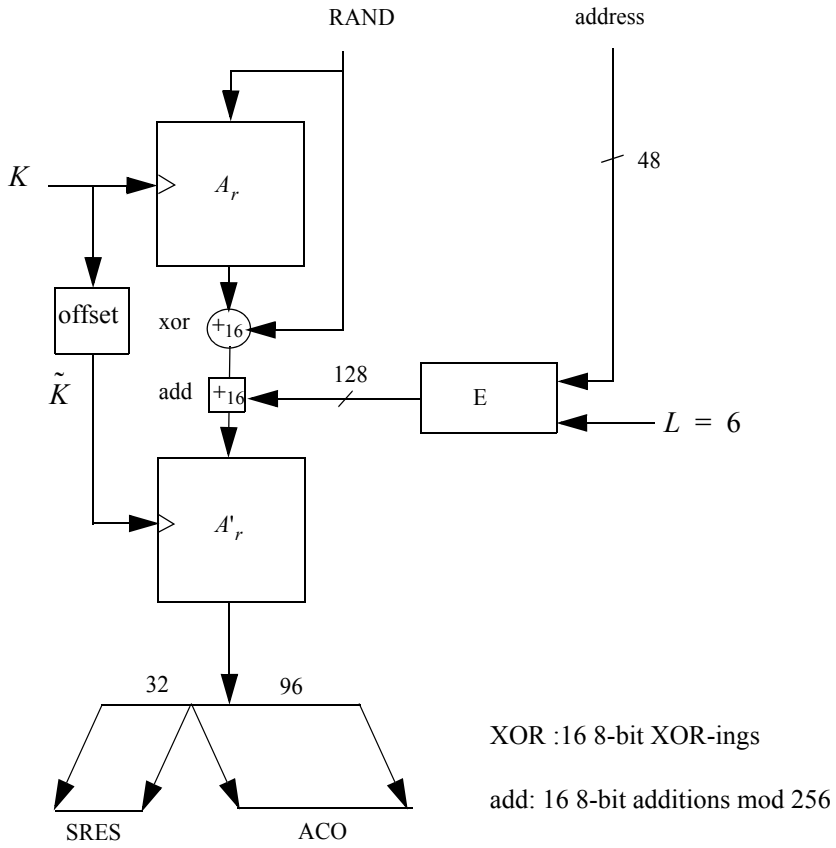


Figure 179—Flow of data for the computation of E_1

13.6.2 The functions A_r and A'_r

The function A_r is identical to SAFER+. It consists of a set of eight layers (each layer is called a *round*) and a parallel mechanism for generating the subkeys $K_p[j]$, $p = 1, 2, \ldots, 17$, which are the round keys to be used in each round. The function will produce a 128-bit result from a 128-bit random input string and a 128-bit key. Besides the function A_r, a slightly modified version referred to as A'_r is used in which the input of round 1 is added to the input of round 3. This is done to make the modified version noninvertible and prevents the use of A'_r (especially in E_{2x}) as an encryption function. See Figure 180 (in 13.6.2.2) for details.

13.6.2.1 The round computations

The computations in each round are a composition of encryption with a round key, substitution, encryption with the next round key, and, finally, a pseudo-Hadamard transform (PHT). The computations in a round shall be as shown in Figure 180 (in 13.6.2.2). The subkeys for round $r, r = 1, 2, \ldots, 8$ are denoted $K_{2r-1}[j]$, $K_{2r}[j]$, $j = 0, 1, \ldots, 15$. After the last round, $K_{17}[j]$ is applied identically to all previous odd numbered keys.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

13.6.2.2 The substitution boxes *e* and *l*

In Figure 180, two boxes are shown, marked *e* and *l*. These boxes implement the same substitutions as are used in SAFER+; i.e., they implement

e, l : $\{0, \ldots, 255\} \rightarrow \{0, \ldots, 255\}$,

e : $i \mapsto (45^i \pmod{257}) \pmod{256}$,

l : $i \mapsto j$ s.t. $i = e(j)$.

Their role, as in the SAFER+ algorithm, is to introduce nonlinearity.

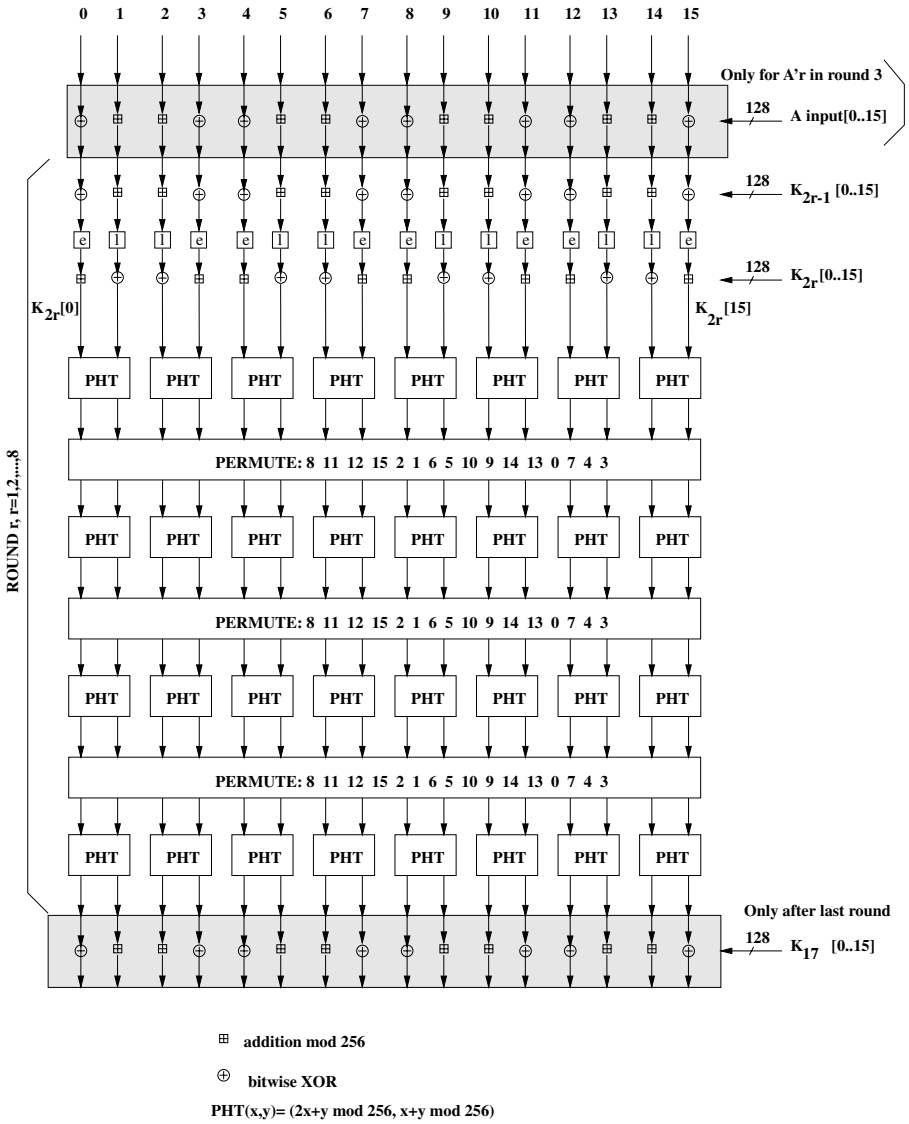


Figure 180—One round in A_r and A'_r

The permutation boxes in the figure show how input byte indices are mapped onto output byte indices. Thus, position 8 is mapped on position 0 (leftmost), position 11 is mapped on position 1, etc.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

13.6.2.3 Key scheduling

In each round, two batches of 16-octet keys are needed. These round keys are derived as specified by the key scheduling in SAFER+. Figure 181 gives an overview of how the round keys $K_p[j]$ are determined. The bias vectors B_2, B_3, ..., B_{17} shall be computed according to Equation (36).

$$B_p[i] = ((45^{(45^{17p+i+1} \bmod 257)} \bmod 257) \bmod 256), \text{ for } i = 0, \ldots, 15. \quad (36)$$

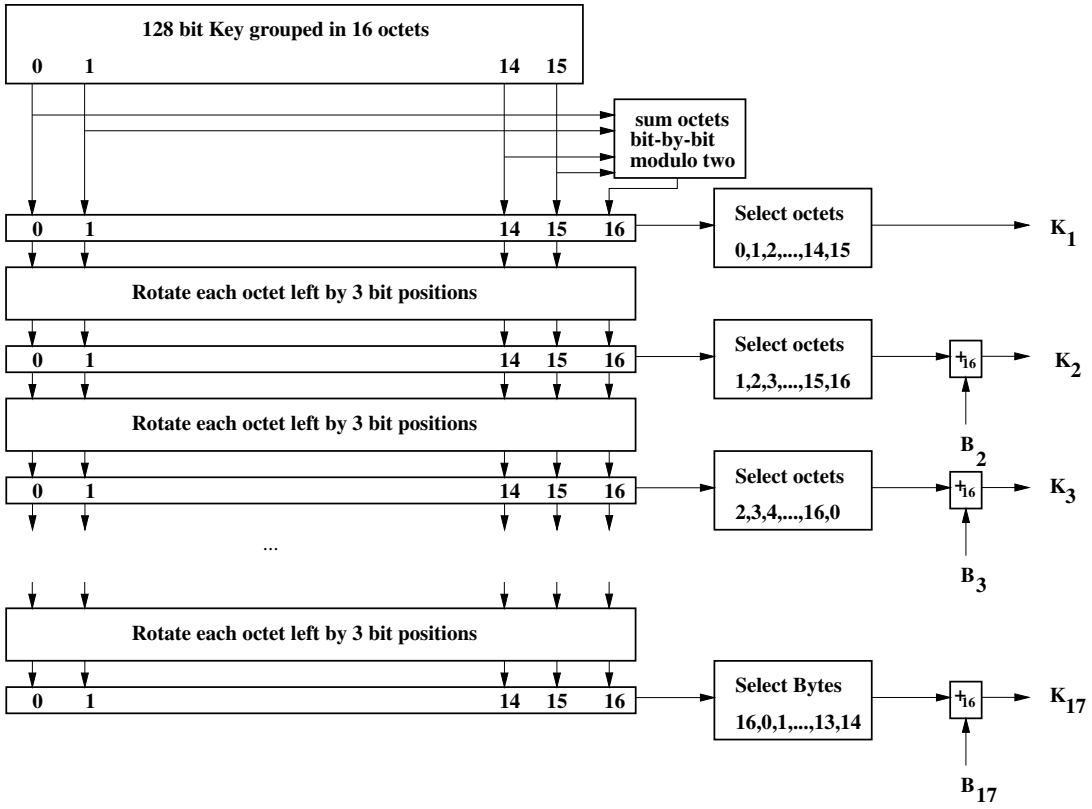


Figure 181—Key scheduling in A_r

13.6.3 E_2-key generation function for authentication

The key used for authentication shall be derived through the procedure that is shown in Figure 182. The figure shows two modes of operation for the algorithm. In the first mode, E_{21} produces a 128-bit link key, K, using a 128-bit RAND value and a 48-bit address. This mode shall be utilized when creating unit keys and combination keys. In the second mode, E_{22} produces a 128-bit link key, K, using a 128-bit RAND value and an L octet user PIN. The second mode shall be used to create the initialization key and also when a master key is to be generated.

When the initialization key is generated, the PIN is augmented with the BD_ADDR (see 13.3.2.1 for which address to use). The augmentation shall always start with the least significant octet of the address immediately following the most significant octet of the PIN. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used.

This key generating algorithm again exploits the cryptographic function. E_2 for mode 1 (denoted E_{21}) is computed according to Equation (37) and Equation (38):

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

$$E_{21}\colon\ \{0,1\}^{128} \times \{0,1\}^{48} \to \{0,1\}^{128}$$
$$(\text{RAND}, \text{address}) \mapsto A'_r(X, Y) \tag{37}$$

where (for mode 1)

$$\begin{cases} X = \text{RAND}[0 \ldots 14] \cup (\text{RAND}[15] \oplus 6) \\ Y = \bigcup_{i=0}^{15} \text{address}[i \ (\text{mod } 6)] \end{cases} \tag{38}$$

Let L be the number of octets in the user PIN. The augmenting is defined by

$$\text{PIN}' = \begin{cases} \text{PIN}[0 \ldots L-1] \cup \text{BD\_ADDR}[0 \ldots \min\{5, 15-L\}], & L < 16, \\ \text{PIN}[0 \ldots L-1], & L = 16, \end{cases} \tag{39}$$

Then, in mode 2, E_2 (denoted E_{22}) is

$$E_{22}\colon\ \{0,1\}^{8L'} \times \{0,1\}^{128} \times \{1, 2, \ldots, 16\} \to \{0,1\}^{128}$$
$$(\text{PIN}', \text{RAND}, \text{L}') \mapsto A'_r(X, Y) \tag{40}$$

where

$$\begin{cases} X = \bigcup_{i=0}^{15} \text{PIN}'[i \ (\text{mod } L')], \\ Y = \text{RAND}[0 \ldots 14] \cup (\text{RAND}[15] \oplus L'), \end{cases} \tag{41}$$

and $L' = \min\{16, L+6\}$ is the number of octets in PIN'.

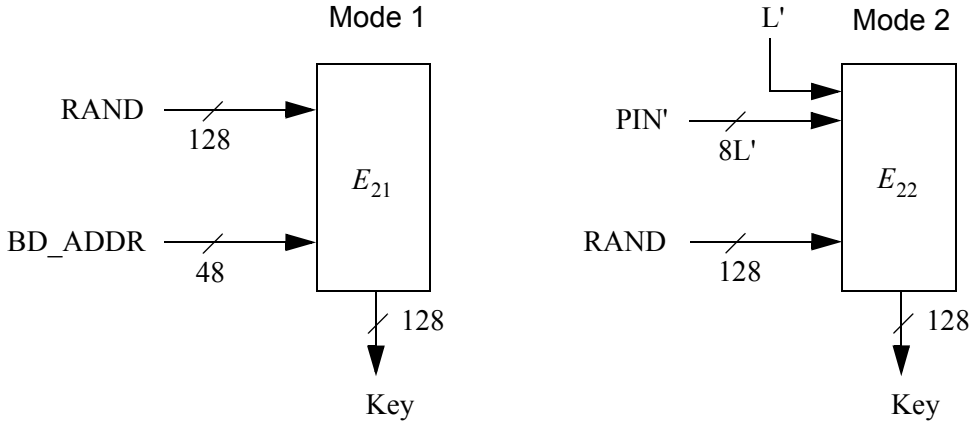


Figure 182—Key generating algorithm E_2 and its two modes

Mode 1 is used for unit and combination keys, while mode 2 is used for K_{init} and K_{master}.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

13.6.4 E_3-key generation function for encryption

The ciphering key K_C used by E_0 shall be generated by E_3. The function E_3 is constructed using A'_r as follows:

$$E_3\colon \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{96} \rightarrow \{0,1\}^{128}$$
$$(K, \text{RAND}, \text{COF}) \mapsto Hash(K, \text{RAND}, \text{COF}, 12) \tag{42}$$

where $Hash$ is the hash function as defined by Equation (32). The key length produced is 128 bits. However, before use within E_0, the encryption key K_C is shortened to the correct encryption key length, as described in 13.4.5. A block scheme of E_3 is depicted in Figure 183.

The value of COF is determined as specified by Equation (22).

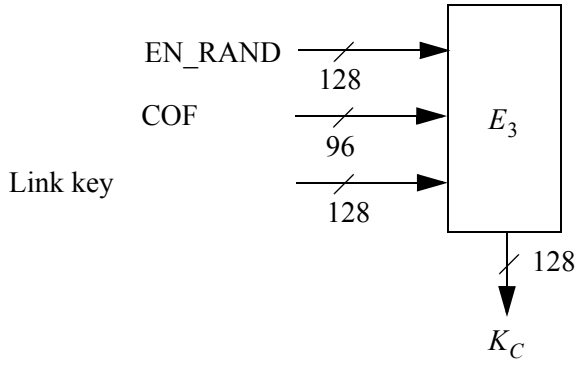


Figure 183—Generation of the encryption key

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14. Logical Link Control and Adaptation Protocol (L2CAP)

The L2CAP supports higher level protocol multiplexing, packet segmentation and reassembly (SAR), and the conveying of QoS information.

This clause defines the L2CAP. It is layered over the LCP and resides in the data link layer (DLL) as shown in Figure 184. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, SAR operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive upper layer data packets (L2CAP SDUs) up to 64 kB in length. L2CAP also permits per-channel flow control and retransmission via the flow control and retransmission modes.

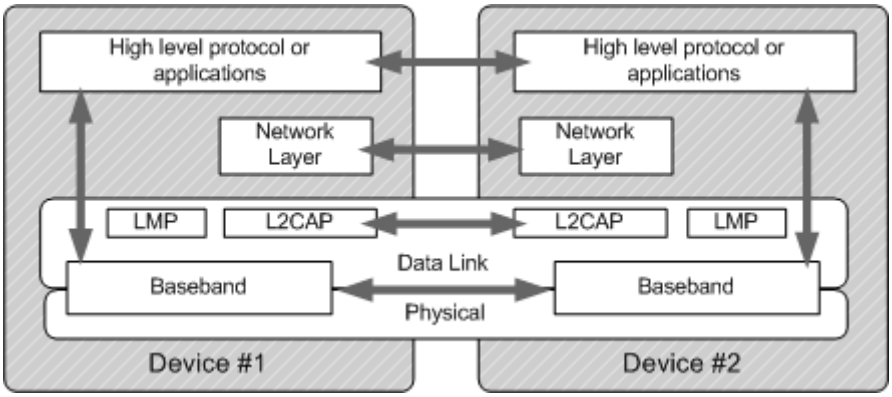


Figure 184—L2CAP within protocol layers

The L2CAP layer provides logical channels, named *L2CAP channels*, which are mapped to L2CAP logical links supported by an ACL logical transport (see 8.4.4).

14.1 L2CAP features

The functional requirements for L2CAP include protocol/channel multiplexing, SAR, per-channel flow control, error control, and group management. Figure 185 illustrates how L2CAP data flows fit into the protocol stack. L2CAP lies above the LCP and interfaces with other communication protocols such as the Bluetooth Service Discovery Protocol (SDP), RFCOMM, Telephony Control Protocol specification (TCS), and Bluetooth Network Encapsulation Protocol (BNEP). Voice-quality channels for audio and telephony applications and synchronous transparent connections are usually run over synchronous logical transports (see 8.4.3). Packetized audio data, such as Internet Protocol (IP) telephony, may be sent using communication protocols running over L2CAP.

Figure 186 breaks down L2CAP into its architectural components. The channel manager provides the control plane functionality and is responsible for all internal signalling, L2CAP peer-to-peer signalling, and signalling with higher and lower layers. It performs the state machine functionality described in 14.6 and uses message formats described in 14.4 and 14.5. The retransmission and flow control block provides per-channel flow control and optional retransmission for applications that require it.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

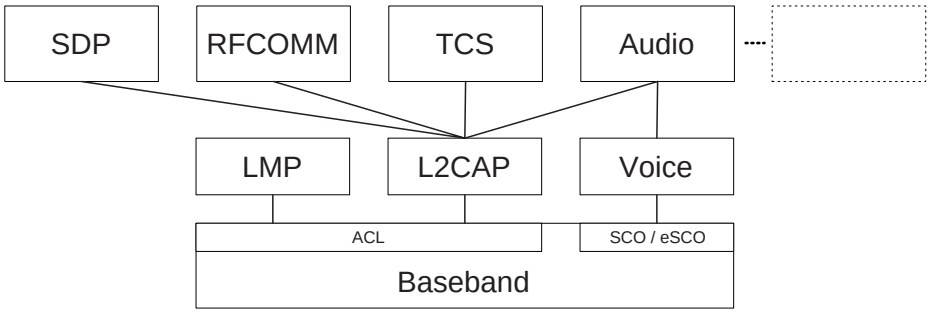


Figure 185—L2CAP data flows in IEEE 802.15.1-2005 architecture

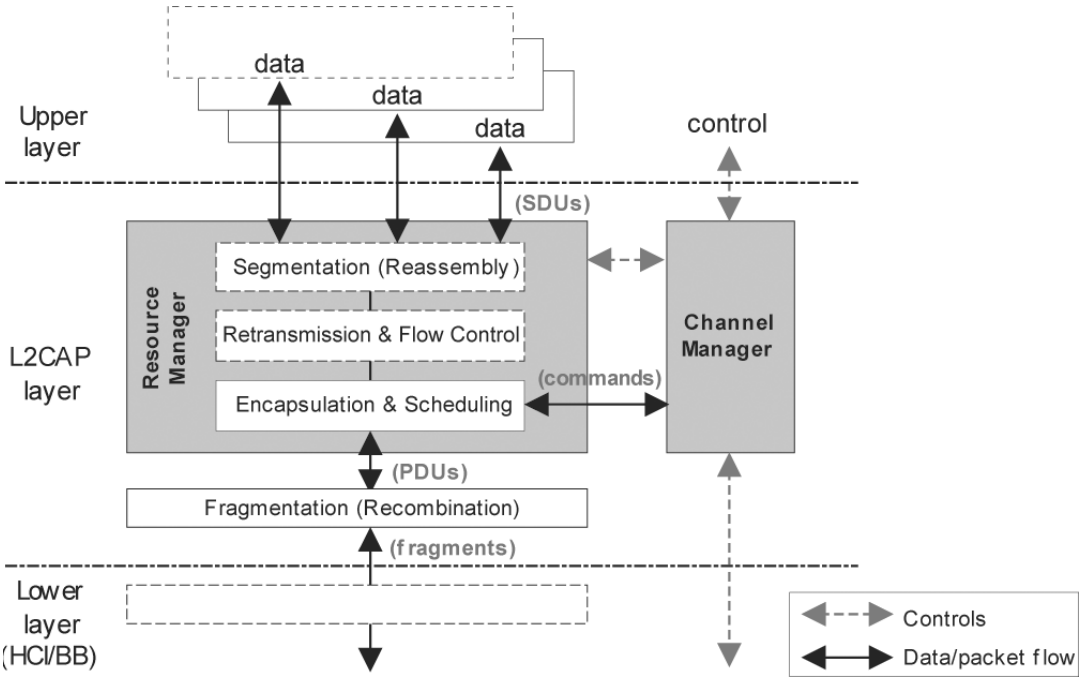


Figure 186—L2CAP architectural blocks

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.

— *Protocol/channel multiplexing.* L2CAP supports multiplexing because the BB 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.

— *SAR.* 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:

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

1) Segmentation will allow the interleaving of application data units in order to satisfy latency requirements.
2) Memory and buffer management is easier when L2CAP controls the packet size.
3) Error correction by retransmission can be made more efficient.
4) 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.
5) 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. Due to the delays between the L2CAP layers, stop-and-go flow control as employed in the BB is not sufficient. A window-based flow control scheme is provided. The use of flow control is an optional aspect of the L2CAP.

— *Error control and retransmissions.* Some applications require a residual error rate much smaller than the BB can deliver. L2CAP includes optional error checks and retransmissions of L2CAP PDUs. The error checking in L2CAP protects against errors due to when the BB falsely accepts packet headers and due to failures of the HEC or CRC error checks on the BB 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.

— *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 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 BB packets. This results in BB packet payloads carrying start and continuation fragments of the L2CAP PDU.

  Each layer of the protocol stack may pass on diffferent 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, and the receiving L2CAP entity still recombines the fragments to obtain the original L2CAP PDU.

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

14.1.1 Assumptions

The protocol is designed based on the following assumptions:

a) The ACL logical transport and L2CAP logical link between two devices is set up using the LMP. The BB provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than one unicast ACL logical transport exists between any two devices.
b) The BB always provides the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bidirectional. Multicasts and unidirectional traffic (e.g., video) do not require duplex channels.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

c) The L2CAP layer provides a channel with a degree of reliability based on the mechanisms available at the BB layer and with optional additional packet segmentation and error detection that can be enabled in the enhanced L2CAP layer. The BB performs data integrity checks and resends data until they have been successfully acknowledged or a timeout occurs. Because acknowledgments may be lost, timeouts may occur even after the data have been successfully sent. The LCP uses a 1-bit SEQN. Note that the use of BB broadcast packets is prohibited if reliability is required, since all broadcasts start the first segment of an L2CAP packet with the same sequence bit.
d) Some applications will expect independent flow control, independence from the effects of other traffic, and, in some cases, better error control than the BB 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.

14.1.2 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 14.3.2.

— L2CAP does not support the concept of a global group name.

14.1.3 Terminology

The formal definitions in Table 493 apply:

Table 493—Terminology

Term	Description
Upper layer	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.
Lower layer	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 controller; however, an HCI may be involved, such that an HCI host driver could also be seen as the lower layer. Except for the HCI (in case HCI is involved), the interface between L2CAP and the lower layer is not specified.
L2CAP channel	The logical connection between two endpoints in peer devices, characterized by their channel identifiers (CIDs), which is multiplexed on the L2CAP logical link, which is supported by an ACL logical transport (see 8.4.2).
SDU, or L2CAP SDU	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.
Segment, or SDU segment	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.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 493—Terminology *(continued)*

Term	Description
Segmentation	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.
Reassembly	The reverse procedure corresponding to segmentation, resulting in an SDU being reestablished 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.
PDU, or L2CAP PDU	A packet of data containing L2CAP 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.
Basic L2CAP header	Minimum L2CAP information that is present in the beginning of each PDU: a length field and a field containing the channel identifier (CID)
Basic information frame (B-frame)	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.
Information frame (I-frame)	A PDU used in the retransmission mode and the flow control mode. It contains an SDU segment and additional protocol information, encapsulated by a basic L2CAP header
Supervisory frame (S-frame)	A PDU used in the retransmission mode and the flow control mode. It contains protocol information only, encapsulated by a basic L2CAP header, and no SDU data.
Control frame (C-frame)	A PDU that contains L2CAP signalling messages exchanged between the peer L2CAP entities. C-frames are exclusively used on the L2CAP signalling channel.
Group frame (G-frame)	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.
Fragment	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.
Fragmentation	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 accomodate the L2CAP PDU transport to HCI data packet or BB packet sizes. Fragmentation of PDUs may be applied in all L2CAP modes. NOTE—In IEEE Std 802.15.1-2002, fragmentation and recombination was referred to as *segmentation and reassembly (SAR)*.
Recombination	The reverse procedure corresponding to fragmentation, resulting in an L2CAP PDU reestablished 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.
Maximum transmission unit (MTU)	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.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 493—Terminology *(continued)*

Term	Description
Maximum PDU payload Size (MPS)	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 MTU is the equivalent to the maximum PDU payload size and shall be made equal in the configuration parameters.
Signalling MTU (MTU_{sig})	The maximum size of command information that the L2CAP layer entity is capable of accepting. The MTU_{sig} refers to the signalling channel only and corresponds to the maximum size of a C-frame, excluding the basic L2CAP header. The MTU_{sig} value of a peer is discovered when a C-frame that is too large is rejected by the peer.
Connectionless MTU (MTU_{cnl})	The maximum size of the connection packet information that the L2CAP layer entity is capable of accepting. The MTU_{cnl} refers to the connectionless channel only and corresponds to the maximum G-frame, excluding the basic L2CAP header. The MTU_{cnl} of a peer can be discovered by sending an information request.
MaxTransmit	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 one 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.

14.2 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).

14.2.1 Channel identifiers (CIDs)

A 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 494 summarizes the definition and partitioning of the CID name space.

Table 494—CID name space

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

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 Table 494). Thus, even if the same CID

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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.

14.2.2 Operation between devices

Figure 187 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID identifies each endpoint of the channel. The connectionless channels restrict data flow to a single direction. These channels are used to support a channel group, where the CID on the source represents one or more remote devices. There are also a number of CIDs reserved for special purposes. The signalling channel is one example of a reserved channel. This channel is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of connection oriented and connectionless channels. Support for a signalling channel within an L2CAP entity is mandatory.

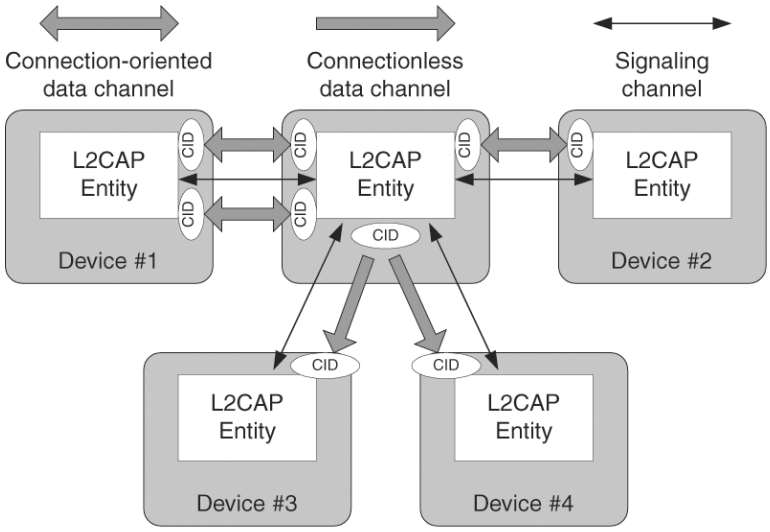


Figure 187—Channels between devices

NOTE—It is assumed that an L2CAP signalling 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 Figure 187, a CID is used to represent a group consisting of device #3 and #4. Traffic sent from this CID is directed to the remote channel reserved for connectionless data traffic.

Table 495 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. The state machine associated with each connection-oriented channel is described in 14.6. The packet format associated with bidirectional channels is described in 14.3.1, and the packet format associated with unidirectional channels is described in 14.3.2.

Table 495—Types of CIDs

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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.2.3 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 standard lists a number of services that should be exported by any L2CAP implementation. Each implementation shall also support a set of signalling commands for use between L2CAP implementations. L2CAP implementations should also be prepared to accept certain types of events from lower layers and generate events to upper layers. How these events are passed between layers is implementation specific. See Figure 188.

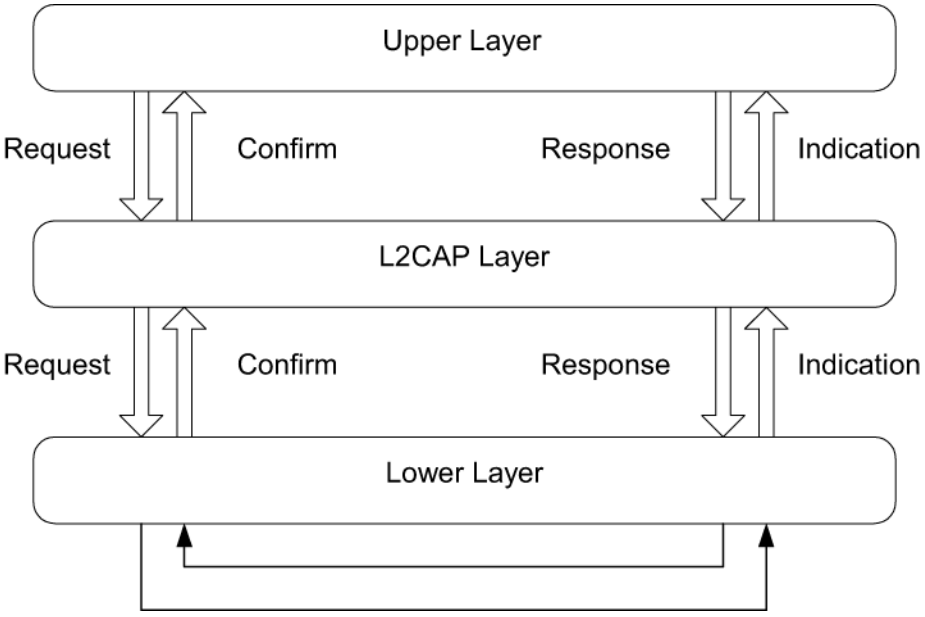


Figure 188—L2CAP transaction model

14.2.4 Modes of operation

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

The modes are as follows:

— Basic L2CAP mode

— Flow control mode

— Retransmission mode

The modes are enabled using the configuration procedure. The basic L2CAP mode is the default mode, which is used when no other mode is agreed.

In flow control and retransmission modes, PDUs exchanged with a peer entity are numbered and acknowledged. The SEQNs in the PDUs are used to control buffering, and a TX window size is used to limit the required buffer space and/or provide a method for flow control. In addition to the window size, the Token Bucket size parameter of the flow specification can be used to dimension the buffers, in particular, on channels that do not use flow and error control.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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 retransmiting PDUs as needed. A mechanism that goes back a specified number of transmissions is used to simplify the protocol and limit the buffering requirements.

14.3 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.

14.3.1 Connection-oriented channel in basic L2CAP mode

Figure 189 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*.

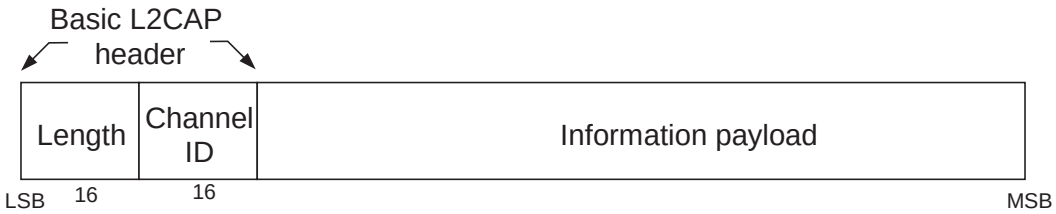


Figure 189—PDU format in basic L2CAP mode on connection-oriented channels (field sizes in bits)

The fields shown are as follows:

— *Length: 2 octets (16 bits)*. Length indicates the size, in octets, of the information payload excluding the length of the L2CAP header. The length of an information payload can be up to 65 535 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 CID identifies the destination channel endpoint of the packet.

— *Information payload: 0 to 65 535 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 14.5.1). The minimum supported MTU for the signalling PDUs (MTU_{sig}) is 48 octets (see 14.4).

14.3.2 Connectionless data channel in basic L2CAP mode

Figure 190 illustrates the L2CAP PDU format within a connectionless data channel. Here, the L2CAP PDU is also referred to as a *G-frame*.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

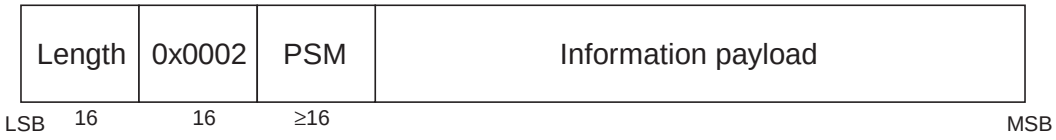


Figure 190—L2CAP PDU format in basic L2CAP mode on connectionless channel

The fields shown are as follows:

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

— *Channel ID: 2 octets.* CID 0x0002 is reserved for connectionless traffic.

— *PSM (protocol/service multiplexer): 2 octets (minimum).* For information on the PSM field, see 14.4.2.

— *Information payload: 0 to 65533 octets.* This contains the payload information to be distributed to all members of the piconet. Implementations shall support a MTU_{cnl} of 48 octets on connectionless channels. Devices may also explicitly change to a larger or smaller MTU_{cnl}.

NOTE—The maximum size of the information payload decreases accordingly if the PSM field is extended beyond the 2-octet minimum.

14.3.3 Connection-oriented channel in retransmission/flow control 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. See Figure 191.

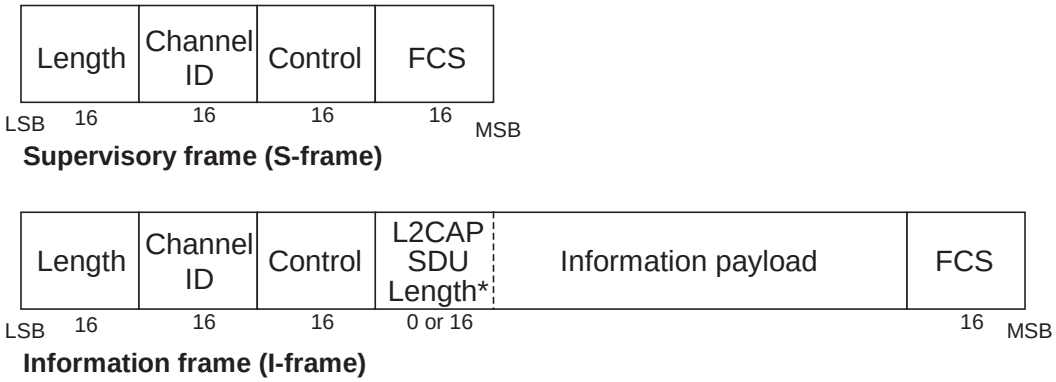


Figure 191—L2CAP PDU formats in flow control and retransmission modes

14.3.3.1 L2CAP header fields

The header fields shown in Figure 191 are as follows:

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

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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

If the L2CAP SDU Length field is present, then the maximum number of information payload octets in one I-frame is 65 529 octets. If the L2CAP SDU Length field is not present, then the maximum number of information payload octets in one I-frame is 65 531 octets.

— *Channel ID: 2 octets.* This field contains the CID.

14.3.3.2 Control field (2 octets)

The Control field identifies the type of frame. The Control field will contain SEQNs where applicable. Its coding is shown in Table 496. There are two different frame types: information (I-frame) and supervisory (S-frame). Information and supervisory frame types are distinguished by the rightmost bit in the Control field, as shown in Table 496, where X denotes reserved bits and shall be coded 0.

Table 496—Control field formats

Frame type	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
I-frame	SAR		ReqSeq						R	TxSeq						0
S-frame	X	X	ReqSeq						R	X	X	X	S		0	1

The I-frames are used to transfer information between L2CAP entities. Each I-frame has a TxSeq (send SEQN) field; ReqSeq (receive SEQN) field, which may or may not acknowledge additional I-frames received by the DLL entity; and a Retransmission Disable (R) bit, which affects whether I-frames are retransmitted. The SAR field in the I-frame is used for SAR control. The L2CAP SDU Length field specifies the length of an SDU, including the aggregate length across all segments if segmented.

The S-frames are used to acknowledge I-frames and request retransmission of I-frames. Each S-frame has an ReqSeq SEQN, which may acknowledge additional I-frames received by the DLL entity, and a Retransmission Disable (R) bit, which affects whether I-frames are retransmitted. Defined types of S-frames are receiver ready (RR) and reject (REJ).

The fields for these frame types are as follows:

— *TxSeq (6 bits).* The send SEQN is used to number each I-frame to enable sequencing and retransmission.

— *ReqSeq (6 bits).* The receive SEQN is used by the receiver side to acknowledge I-frames and, in the REJ S-frame, to request the retransmission of an I-frame with a specific send SEQN.

— *R (1 bit).* The R 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 R bit set, it shall disable the retransmission timer. This causes the sender to stop retransmitting I-frames.

  R = 0: Normal operation. Sender uses the retransmission timer to control retransmission of I-frames. Sender does not use the monitor timer.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

R = 1: Receiver side requests sender to postpone retransmission of I-frames. Sender monitors signalling with the monitor timer. Sender does not use the retransmission timer.

The functions of the ReqSeq and R fields are independent.

— *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 = 01 (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 497.

Table 497—SAR control element format

Code	Description
00	Unsegmented L2CAP SDU
01	Start of L2CAP SDU
10	End of L2CAP SDU
11	Continuation of L2CAP SDU

— *S (supervisory function) (2 bits).* The S bits mark the type of S-frame. There are two types defined: RR and REJ. The encoding is shown in Table 498.

Table 498—S control element format: type of S-frame

Code	Description
00	RR (receiver ready)
01	REJ (reject)
10	Reserved
11	Reserved

14.3.3.3 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.

14.3.3.4 Information payload (0–65 531 octets)

The information payload 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. For I-frames without an SDU Length field, this limits the maximum number of octets in the field to 65 531. For I-frames with an SDU Length field (SAR = 01), this limits the maximum number of octets in the field to 65 529. Thus, even

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

if an MPS of 65 531 has been negotiated, the range of the basic L2CAP header Length field will restrict the number of octets in this field when an SDU Length field is present to 65 529.

14.3.3.5 FCS (2 octets)

The FCS field is 2 octets. The FCS is constructed using the generator polynomial $g(D) = D^{16} + D^{15} + D^2 + 1$ (see Figure 192). The 16-bit LFSR is initially loaded with the value 0x0000, as depicted in Figure 193. The switch S is set in position 1 while data are 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, and so on). The FCS covers the basic L2CAP header, Control field, L2CAP SDU Length field, and information payload, if present, as shown in Figure 191 (in 14.3.3.1).

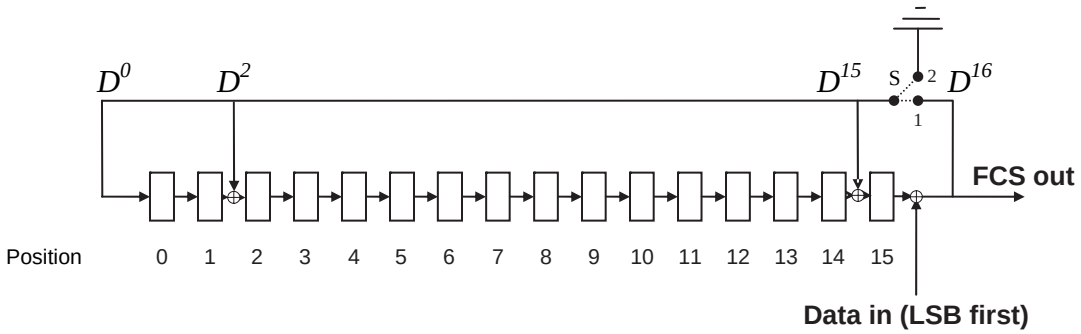


Figure 192—The LFSR circuit generating the FCS

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LFSR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 193—Initial state of the FCS generating circuit

Examples of FCS calculation, $g(D) = D^{16} + D^{15} + D^2 + 1$, are as follows:

— ***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 00 01 02 03 04 05 06 07 08 09 38 61 (hexadecimal notation)

— ***RR S-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)

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.3.3.6 Invalid frame detection

A received PDU shall be regarded as invalid if one of the following conditions occurs:

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

These error conditions may be used for error reporting.

14.4 Signalling packet formats

This subclause describes the signalling commands passed between two L2CAP entities on peer devices. All signalling commands are sent to the signalling channel with CID 0x0001. This signalling channel is available as soon as an ACL logical transport is set up and L2CAP traffic is enabled on the L2CAP logical link. Figure 194 illustrates the general format of L2CAP PDUs containing signalling 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 MTU_{sig}. The minimum supported payload length for the C-frame (MTU_{sig}) is 48 octets. L2CAP implementations should not use C-frames that exceed the MTU_{sig} of the peer device. If they ever do, the peer device shall send a Command Reject packet (defined in 14.4.1) containing the supported MTU_{sig}. Implementations must be able to handle the reception of multiple commands in an L2CAP packet.

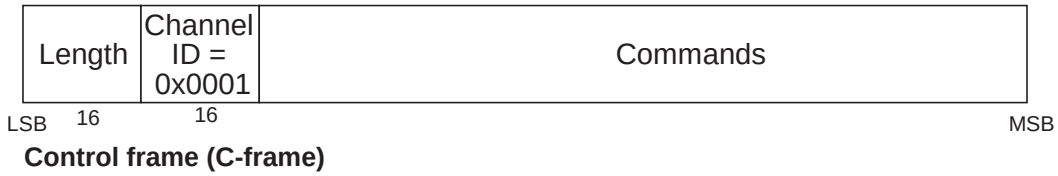


Figure 194—L2CAP PDU format on the signalling channel

Figure 195 displays the general format of all signalling commands.

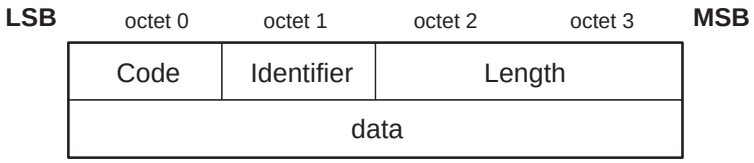


Figure 195—Command format

The fields shown are as follows:

— *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 is sent in response.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 499 lists the codes defined by this standard. All codes are specified with the MSB in the left-most position.

Table 499—Signalling command codes

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

— *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 endpoint is not responding to signalling requests. On the expiration of a RTX or ERTX timer, the same identifier shall be used if a duplicate request is resent as stated in 14.6.2.

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.

14.4.1 Command Reject packet (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 196 displays the format of the packet. The CID shall match the CID of the command packet being rejected. Implementations shall always send these packets in response to unidentified signalling packets. Command Reject packets should not be sent in response to an identified response packet.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

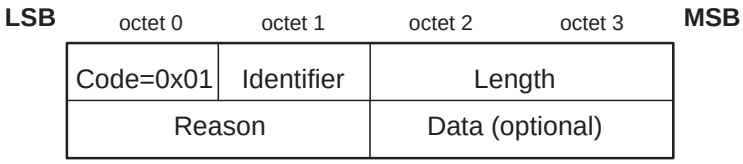


Figure 196—Command Reject packet

When multiple commands are included in an L2CAP packet and the packet exceeds the MTU_{sig} of the receiver, a single Command Reject packet shall be sent in response. The CID shall match the first request command in the L2CAP packet. If only responses are recognized, the packet shall be silently discarded.

The data fields are as follows:

— *Reason (2 octets)*. The Reason field describes why the request packet was rejected and is set to one of the reason codes in Table 500.

Table 500—Command Reject packet reason code descriptions

Reason value	Description
0x0000	Command not understood
0x0001	MTU_{sig} exceeded
0x0002	Invalid CID in request
Other	Reserved

— *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 (MTU_{sig} exceeded), the 2-octet Data field represents the maximum MTU_{sig} 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 packet) 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 501—Command Reject packet reason code data values

Reason code value	Data length (octets)	Data value
0x0000	0	N/A
0x0001	2	Actual MTU_{sig}
0x0002	4	Requested CID

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.4.2 Connection Request packets (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 197 illustrates a Connection Request packet.

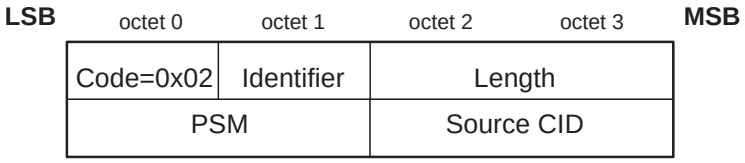


Figure 197—Connection Request packet

The data fields are as follows:

— *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/IEC 3309 extension mechanism for address fields. All PSM values shall be odd, i.e., the LSB of the least significant octet must be 1. Also, all PSM values shall have the LSB 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 SDP. The dynamically assigned values may be used to support multiple implementations of a particular protocol.

Table 502—Connection Request packet defined PSM values[a]

PSM value	Description
0x0001	SDP
0x0003	RFCOMM
0x0005	Telephony Control Protocol
0x0007	TCS cordless
0x000F	BNEP
0x0011	HID Control
0x0013	HID Interrupt
0x0015	UPnP (ESDP)
0x0017	AVCTP
0x0019	AVDTP
Other < 1000	Reserved
[0x1001-0xFFFF]	Dynamically assigned

[a]The most recent PSM assignments can be found in the Bluetooth Assigned Numbers [B1] database.

— *Source CID (2 octets)*. The source CID is 2 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.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.4.3 Connection Response packet (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 198.

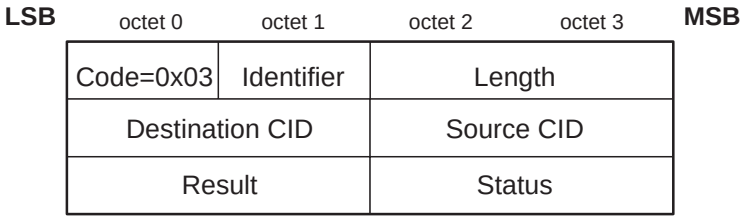


Figure 198—Connection Response packet

The data fields are as follows:

— *Destination CID (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 CID (2 octets).* This field contains the channel endpoint on the device receiving this response packet. This is copied from the Source CID 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 nonzero value indicates the connection request failed or is pending. A logical channel is established on the receipt of a successful result. Table 503 defines values for this field. The Destination CID and Source CID fields shall be ignored when the Result field indicates the connection was refused.

Table 503—Configuration Response packet result values

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

— *Status (2 octets).* The status is defined only when the Result field = 0x0001 (Connection pending) and indicates the status of the connection. The status is set to one of the values shown in Table 504.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 504—Configuration Response packet status values

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

14.4.4 Configuration Request packet (code 0x04)

Configuration Request packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to renegotiate this contract whenever appropriate. During a renegotiation session, all data traffic on the channel should be suspended pending the outcome of the negotiation. Each configuration parameter in a Configuration Request packet shall be related exclusively to either the outgoing or the incoming data traffic, but not to both of them. In 14.5, the various configuration parameters and their relation to the outgoing or incoming data traffic are shown. If an L2CAP entity receives a Configuration Request packet while it is waiting for a response, it shall not block sending the Configuration Response packet; 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 standard 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 nondefault parameters the device sending the Configuration Request packet will accept. The configuration request cannot 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 that the remote device will accept, then it must wait for a Configuration Request packet containing that configuration parameter to be sent from the remote device.

See 14.7.1 for details of the configuration procedure.

Figure 199 defines the format of the Configuration Request packet.

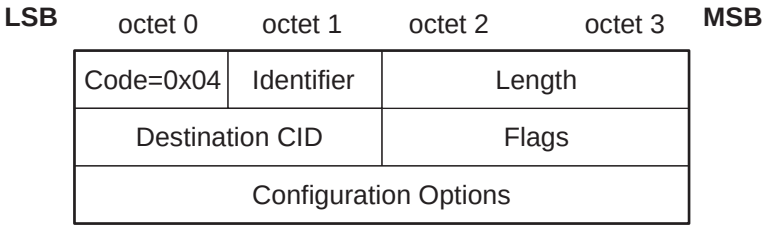


Figure 199—Configuration Request packet

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The data fields are as follows:

— *Destination CID (2 octets).* This field contains the channel endpoint on the device receiving this request packet.

— *Flags (2 octets).* Figure 200 shows the two-octet Flags field. Note that the MSB is shown on the left.

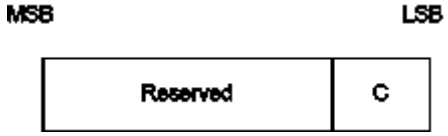


Figure 200—Configuration Request packet Flags field format

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

When all configuration options cannot fit into a Configuration Request packet 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 packet with the Continuation flag set to zero. Each Configuration Request packet 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 CID and shall be matched with a response with the same CID.

When used in the Configuration Request packet, 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 packet with a Configuration Response packet containing the same option(s) present in the request (except for those error conditions more appropriate for a Command Reject packet), or the responder may reply with a Success Configuration Response packet containing no options, thereby 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 14.5. A Configuration Request packet may contain no options (referred to as an *empty* or *null* Configuration Request packet) and can be used to request a response. For an empty Configuration Request packet, the Length field is set to 0x0004.

14.4.5 Configuration Response packet (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 packet response. Each configuration parameter value (if any is present) in a Configuration Response packet reflects an "adjustment" to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding Configuration Request

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

packet. 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 cannot adjust the value in the reverse direction.

The options sent in the response depend on the value in the Result field. Figure 201 defines the format of the Configuration Response packet. See also 14.7.1 for details of the configuration process.

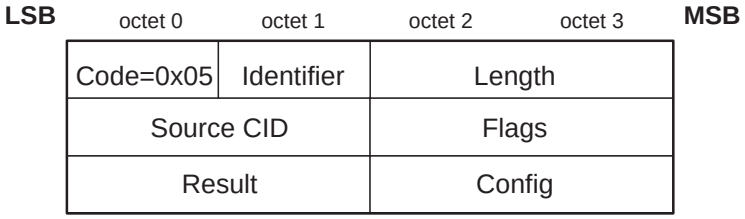


Figure 201—Configuration Response packet

The data fields are as follows:

— *Source CID (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 Source CID matches its local CID paired with the original Destination CID.

— *Flags (2 octets).* Figure 202 displays the 2-octet Flags field. Note that the MSB is shown on the left.

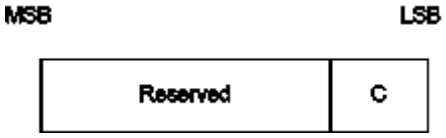


Figure 202—Configuration Response Flags field format

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 the request was acceptable. See Table 505 for possible result codes.

Table 505—Configuration Response packet result codes

Result	Description
0x0000	Success
0x0001	Failure – unacceptable parameters
0x0002	Failure – rejected (no reason provided)

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 505—Configuration Response packet result codes *(continued)*

Result	Description
0x0003	Failure – unknown options
Other	Reserved

— *Config.* This field contains the list of parameters being configured. These are defined in 14.5. On a successful result, these parameters contain the return values for any wild card parameter values (see 14.5.3) 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 packet if they need to be changed.

Each configuration parameter is one-directional. The configuration parameters describe the nondefault parameters the device sending the Configuration Request packetwill accept. The Configuration Request packet cannot 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 packet 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 14.5). 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.

14.4.6 Disconnection Request packet (code 0x06)

Terminating an L2CAP channel requires that a Disconnection Request packet be sent and acknowledged by a Disconnection Response packet. Figure 203 shows a Disconnection Request packet. The receiver shall ensure that both source and destination CIDs match before initiating a channel disconnection.

Once a Disconnection Request packet 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 packet for a channel has been received, all data queued to be sent out on that channel shall be discarded.

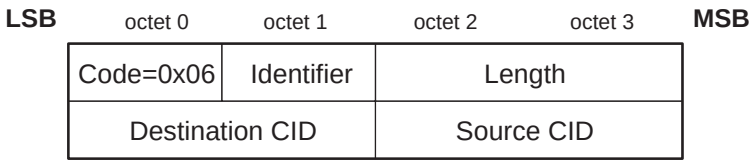


Figure 203—Disconnection Request packet

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The data fields are as follows:

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

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

The Source and Destination CIDs are relative to the sender of this request and shall match those of the channel to be disconnected. If the Destination CID is not recognized by the receiver of this message, a Command Reject message with reason code *invalid CID in request* shall be sent in response. If the receiver finds a Destination CID match, but the Source CID fails to find the same match, the request should be silently discarded.

14.4.7 Disconnection Response packet (code 0x07)

Disconnection responses shall be sent in response to each valid Disconnection Request packet. See Figure 204.

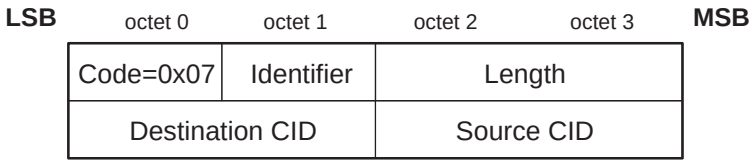


Figure 204—Disconnection Response packet

The data fields are as follows:

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

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

The Destination and Source CID fields (which are relative to the sender of the request) and the Identifier field shall match those of the corresponding Disconnection Request command. If the CIDs do not match, the response should be silently discarded at the receiver.

14.4.8 Echo Request packet (code 0x08)

Echo Request packets 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. See Figure 205.

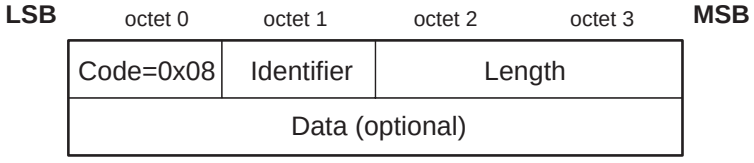


Figure 205—Echo Request packet

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.4.9 Echo Response packet (code 0x09)

An Echo Response packet shall be sent upon receiving a valid Echo Request packet. The CID in the response shall match the CID 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. See Figure 206.

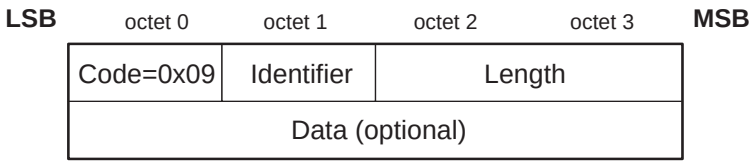


Figure 206—Echo Response packet

14.4.10 Information Request packet (code 0x0a)

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

An L2CAP implementation shall use only optional features or attribute ranges for which the remote L2CAP entity has indicated support through an Information Response packet. Until an Information Response packet that indicates support for optional features or ranges has been received, only mandatory features and ranges shall be used. See Figure 207.

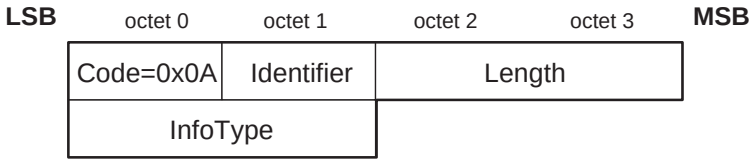


Figure 207—Information Request packet

The data fields are as follows:

— *InfoType (2 octets)*. The InfoType field defines the type of implementation-specific information being requested (see Table 506). See 14.4.11 for details on the type of information requested.

Table 506—InfoType field value definitions

Value	Description
0x0001	MTU_{cnl}
0x0002	Extended features supported
Other	Reserved

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.4.11 Information Response packet (code 0x0b)

An Information Response packet shall be sent upon receiving a valid Information Request packet. The CID in the response shall match the CID 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 field value is not supported. See Figure 208.

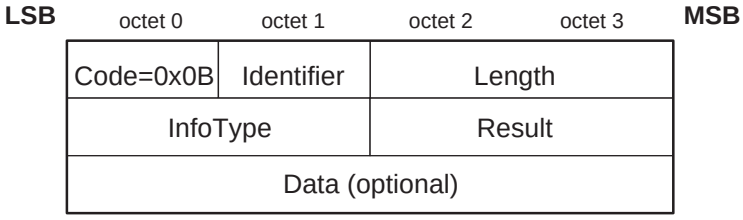


Figure 208—Information Response packet

The data fields are as follows:

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

— *Result (2 octets)*. The Result field contains information about the success of the request. If Result = 0x0000 (Success), the Data field contains the information as specified in Table 508. If Result = 0x0001 (Not supported), no data shall be returned.

Table 507—Information Response packet result values

Value	Description
0x0000	Success
0x0001	Not supported
Other	Reserved

— *Data (0 or more octets)*. The contents of the Data field depend on the InfoType field. See Table 508.

For InfoType = 0x0001, the data field contains the remote entity's 2-octet acceptable MTU_{cnl}. The default value is defined in 14.3.2.

For InfoType = 0x0002, the data field contains the 4-octet L2CAP extended features mask. The features mask refers to the extended features that the L2CAP entity sending the Information Response packet supports. The feature bits contained in the L2CAP features mask are specified in 14.4.12.

L2CAP entities of versions prior to IEEE Std 802.15.1-2005, receiving an Information Request packet with InfoType = 0x0002 for an L2CAP feature discovery, will return an Information Response packet with result code *not supported*.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 508—Information Response packet Data field values

InfoType	Data	Data length (octets)
0x0001	MTU_{cnl}	2
0x0002	Extended features mask	4

14.4.12 Extended features mask

The features are represented as a bit mask in the Information Response packet's Data field (see 14.4.11). For each feature, a single bit is specified that 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 features mask shown in Table 509 consists of 4 octets (numbered octet 0 ... 3), with bit numbers 0 ... 7 each. Within the Information Response packet's Data field, bit 0 of octet 0 is aligned leftmost, bit 7 of octet 3 is aligned rightmost.

NOTE—The L2CAP features mask is a new concept introduced in IEEE Std 802.15.1-2005 and thus contains new features introduced after IEEE Std 802.15.1-2002.

Table 509—Extended features mask

No.	Supported feature	Octet	Bit
0	Flow control mode	0	0
1	Retransmission mode	0	1
2	Bidirectional QoS[a]	0	2
31	Reserved for features mask extension	3	7

[a]Peer side supports upper layer control of the LM's bidirectional QoS. See 14.5.3 for more details.

14.5 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 209 illustrates the format of an option.

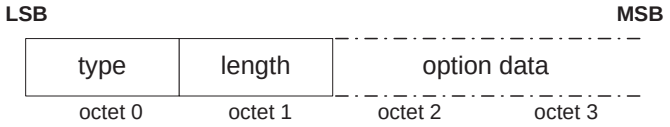


Figure 209—Configuration option format

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The configuration option fields are as follows:

— *Type (1 octet).* The Type field defines the parameters being configured. The MSB 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 packet.

  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.

14.5.1 MTU option

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 2-octet MTU size value as the only information element (see Figure 210). 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 FCS fields as well as the information payload octets.

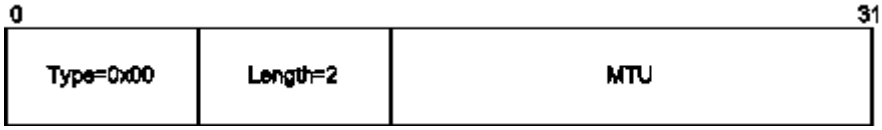


Figure 210—MTU option format

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 affected only by the profile directly using the channel. For example, if a service discovery (SD) transaction is initiated by a non-SD profile, that profile does not affect the MTU of the L2CAP channel used for SD.

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 signalling described in 14.4.5 may be used to reject an MTU smaller than the minimum MTU for a channel. The result code *failure – unacceptable parameters* in the Configuration Response packet 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 packet, 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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Configuration Request packet and the outgoing MTU capability of the device sending the Configuration Response packet. The new agreed value (the default value in a future reconfiguration) 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 packet as explained in 14.4.4.

The option data field is as follows:

— *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.[17]

14.5.2 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 8.7.6.3. 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, and other channels on the same ACL logical transport may, therefore, have other values. See Figure 211.

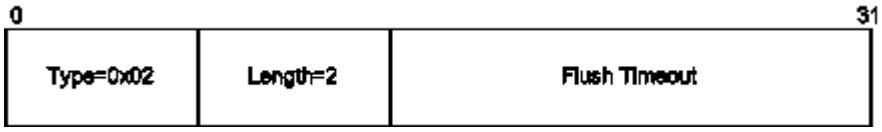


Figure 211—Flush timeout option format

The option data field is as follows:

— *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 as follows:

0x0001 - No retransmissions at the BB level should be performed since the minimum polling interval is 1.25 ms.

0x0002 to 0xFFFE - Flush timeout used by the BB.

0xFFFF - An infinite amount of retransmissions. This is also referred to as a *reliable channel*. In this case, the BB shall continue retransmissions until physical link loss is declared by LM timeouts.

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

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.5.3 QoS option

This option specifies a flow specification similar to RFC 1363. Although the RFC 1363 flow specification addresses only the transmit characteristics, the 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 packet, this option describes the outgoing traffic flow from the device sending the request. In a positive Configuration Response packet, this option describes the incoming traffic flow agreement to the device sending the response. In a negative Configuration Response packet, this option describes the preferred incoming traffic flow to the device sending the response.

L2CAP implementations are required to support only 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 packet contains information that depends on the value of the Result field (see 14.4.5). 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 in the list of option data fields in this subclause) contained in the request. For the result code *failure – unacceptable parameters*, the response shall include a list of outgoing flow specification parameters and parameter values that would make a new Connection Request packet from the local device acceptable by the remote device. Configuration parameters both explicitly referenced in a Configuration Request packet or implied can be included in a Configuration Response packet. Recall that any missing configuration parameters from a Configuration Request packet are assumed to have their most recently accepted values.

If a Configuration Request packet 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. See Figure 212.

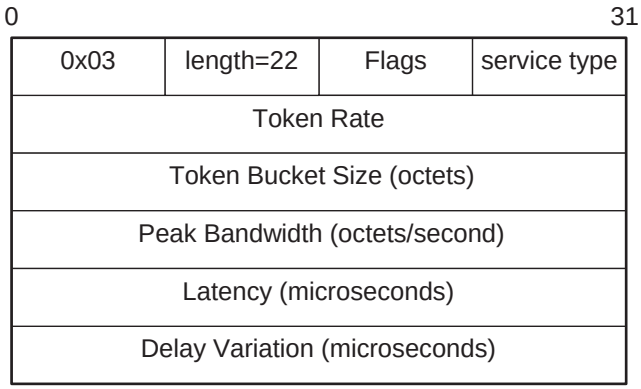


Figure 212—QoS option format containing flow specification

The option data fields are as follows:

— *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 510 defines the different services available. The default value is best effort.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

If best effort service 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 service is selected, the remainder of the fields shall be ignored because there are no data being sent across the channel in the outgoing direction.

Table 510—Service type definitions

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

— *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 field discussions below in this list). 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 overhead. The token rate signalled over the interface between L2CAP and the LM shall include the L2CAP overhead. Furthermore, the Token Rate field 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 are transmitted by the application. When traffic is transmitted and there are insufficient credits available, the traffic is nonconformant. The QoS guarantees are provided only for conformant traffic. For nonconformant traffic, there may not be sufficient resources such as bandwidth and buffer space. Furthermore nonconformant 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 service, 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 field 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 field discussion below in this list). 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 overhead. The token bucket size signalled over the interface between L2CAP and LM shall include the L2CAP overhead. Furthermore, the Token Bucket Size field value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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 service, 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 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 overhead. The peak bandwidth signalled over the interface between L2CAP and LM shall include the L2CAP overhead. Furthermore, the Peak Bandwidth field value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

The value of 0x00000000 means “do not 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 LM, 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 field value may be different when signalled between L2CAP and the LM to account for any queuing delay at the L2CAP transmit side. Furthermore, the Access Latency field 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.

14.5.4 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.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

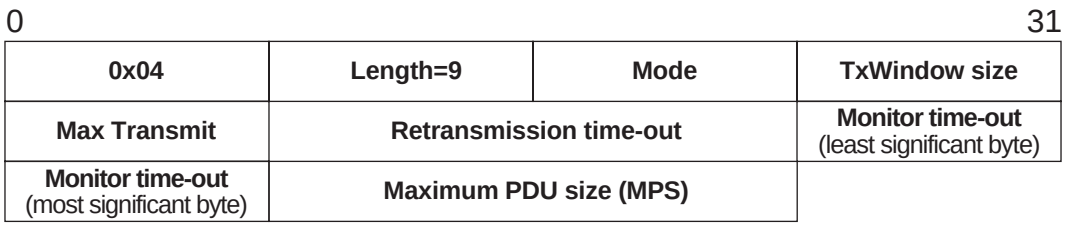


Figure 213—Retransmission and flow control option format

The option data fields are as follows:

— *Mode (1 octet)*. The field contain the requested mode of the link. Possible values are shown in Table 511.

Table 511—Mode definitions

Value	Description
0x00	Basic L2CAP mode
0x01	Retransmission mode
0x02	Flow control mode
Other values	Reserved for future use

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

Retransmission mode should be enabled if a reliable channel has been requested or if the L2CAP flush timeout is long enough to contain the round-trip delay of a retransmission request.

— *TxWindow size (1 octet)*. This field specifies the size of the transmission window for flow control mode and retransmission mode. The range is 1 to 32.

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.

— *MaxTransmit (1 octet)*. This field controls the number of transmissions of a single I-frame that L2CAP is allowed to try in retransmission mode. The minimum value is 1 (one transmission is permitted).

The MaxTransmit field value controls the number of retransmissions that L2CAP is allowed to try in retransmission mode before accepting that a packet and the link is lost. 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 link will be dropped as soon as a packet is lost. The MaxTransmit field shall not be set to zero.

— *Retransmission timeout (2 octets)*. This is the value, in milliseconds, of the retransmission timeout (this value is used to initialize the retransmission timer).

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

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

should be set high enough to avoid unnecessary retransmissions due to delayed acknowledgments. Suitable values could be 100s of milliseconds and up.

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

— *Monitor timeout (2 octets)*. 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 monitor timer).

The monitor timer ensures that lost acknowledgments are retransmitted. Its main use is to recover Retransmission Disable bit changes in lost frames when no data are 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 timeout should also be set to 100s of milliseconds or higher.

— *MPS (2 octets)*. The maximum size, in octets, of payload data that the L2CAP layer entity is capable of accepting, i.e., the MPS corresponds to the maximum PDU payload size.

The settings are configured separately for the two directions of an L2CAP connection. For example, 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 occurs only during configuration.

14.6 State machine

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

14.6.1 General rules for the state machine

— It is implementation specific, and outside the scope of this specification, how the transmissions are triggered.

— The term *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 14.6.1.3).

— OPEN – User data transfer state.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— 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 14.3, including PDUs containing B-frames, I-frames, and S-frames.

Some state transitions and actions are triggered only by internal events effecting one of the L2CAP entity implementations, not by preceding L2CAP signalling 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 message 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 14.6.2 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 subclause.

The states in 14.6.1.1 through 14.6.1.6 and transitions are illustrated in Figure 214.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

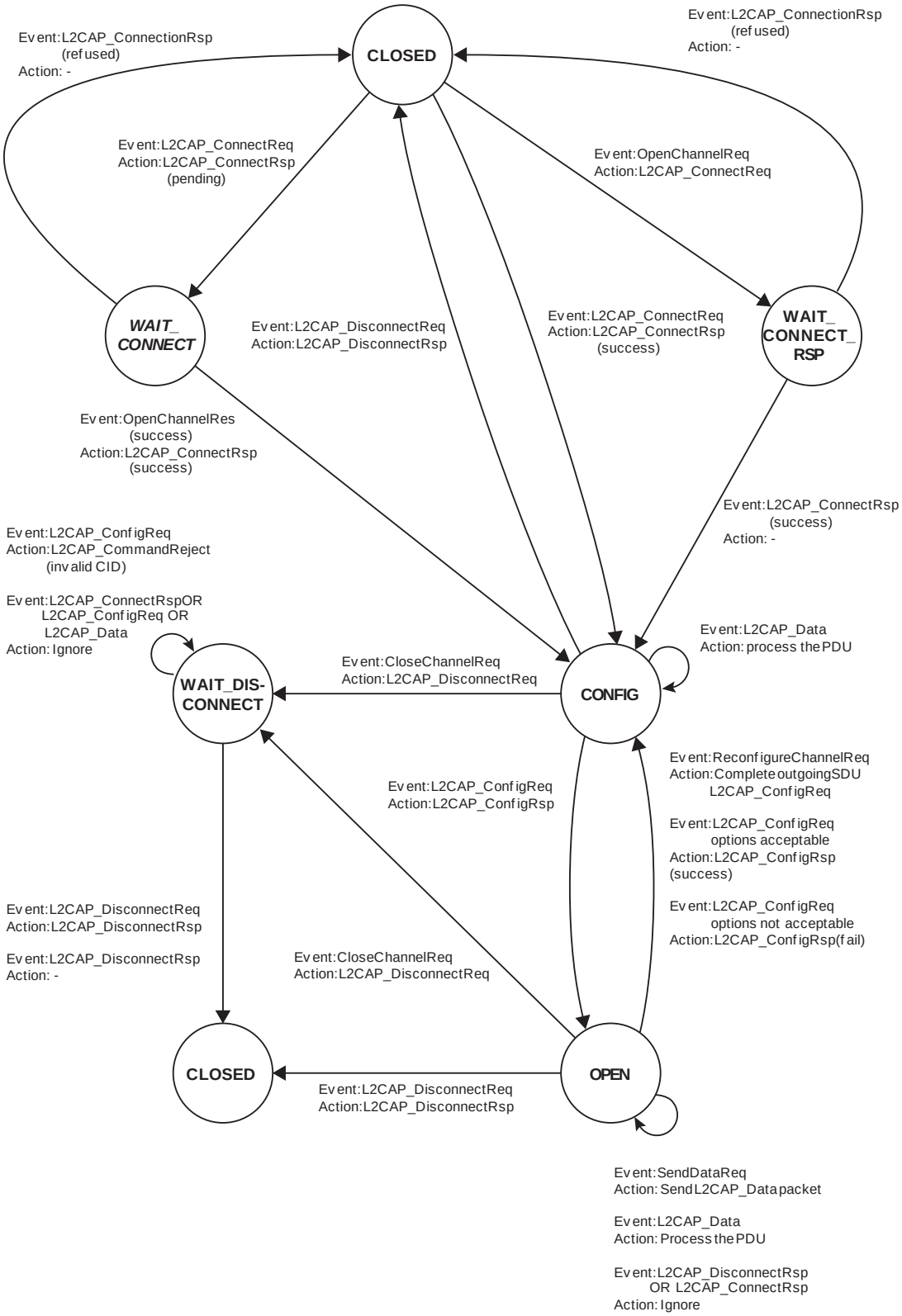


Figure 214—States and transitions

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.6.1.1 CLOSED state

See Table 512 for the events associated with the CLOSED state.

Table 512—CLOSED state events

Event	Condition	Action	Next state
OpenChannel_req	—	Send L2CAP_ConnectReq	WAIT_CONNECT_RSP
L2CAP_ConnectReq	Normal, connection is possible	Send L2CAP_ConnectRsp (success)	CONFIG (substate WAIT_CONFIG)
L2CAP_ConnectReq	Need to indicate pending	Send L2CAP_ConnectRsp (pending)	WAIT_CONNECT
L2CAP_ConnectReq	No resource, not approved, etc	Send L2CAP_ConnectRsp (refused)	CLOSED
L2CAP_ConnectRsp	—	Ignore	CLOSED
L2CAP_ConfigReq	—	Send L2CAP_CommandReject (with reason Invalid CID)	CLOSED
L2CAP_ConfigRsp	—	Ignore	CLOSED
L2CAP_DisconnectReq	—	Send L2CAP_DisconnectRsp	CLOSED
L2CAP_DisconnectRsp	—	Ignore	CLOSED
L2CAP_Data	—	Ignore	CLOSED

NOTE—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 and thus the branch into a new instance of the state machine.

14.6.1.2 WAIT_CONNECT_RSP state

See Table 513 for the events associated with the WAIT_CONNECT_RSP state.

Table 513—WAIT_CONNECT_RSP state events

Event	Condition	Action	Next state
L2CAP_ConnectRsp	Success indicated in result	Send L2CAP_ConfigReq	CONFIG (substate WAIT_CONFIG)
L2CAP_ConnectRsp	Result pending	—	WAIT_CONNECT_RSP
L2CAP_ConnectRsp	Remote side refuses connection	—	CLOSED
L2CAP_ConfigReq	—	Send L2CAP_CommandReject (with reason Invalid CID)	WAIT_CONNECT_RSP

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 513—WAIT_CONNECT_RSP state events *(continued)*

Event	Condition	Action	Next state
L2CAP_ConfigRsp	—	Ignore	WAIT_CONNECT_RSP
L2CAP_DisconnectRsp	—	Ignore	WAIT_CONNECT_RSP
L2CAP_Data	—	Ignore	WAIT_CONNECT_RSP

NOTE—An L2CAP_DisconnectReq message is not included in Table 513 since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

14.6.1.3 WAIT_CONNECT state

See Table 514 for the events associated with the WAIT_CONNECT state.

Table 514—WAIT_CONNECT state events

Event	Condition	Action	Next state
OpenChannel_Rsp	Pending connection request is finally acceptable	Send L2CAP_Connect_Rsp (success)	CONFIG (substate WAIT_CONFIG)
OpenChannel_Rsp	Pending connection request is finally refused	Send L2CAP_Connect_Rsp (refused)	CLOSED
L2CAP_ConnectRsp	—	Ignore	WAIT_CONNECT
L2CAP_ConfigRsp	—	Ignore	WAIT_CONNECT
L2CAP_DisconnectRsp	—	Ignore	WAIT_CONNECT
L2CAP_Data	—	Ignore	WAIT_CONNECT

NOTE—An L2CAP_DisconnectReq or L2CAP_ConfigReq message is not included in Table 514 since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

14.6.1.4 CONFIG state

As it is also described in 14.7.1, both L2CAP entities initiate a Configuration Request packet during the configuration process. This means that each device adopts an initiator role for the outgoing Configuration Request packet and an acceptor role for the incoming Configuration Request packet. 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 packet, but has neither initiated a Configuration Request packet yet nor received a Configuration Request packet with acceptable parameters.

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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— 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 14.6.1.1 and 14.6.1.2, 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 515 shows the success transitions where transitions on one of the minimum paths (no previous nonsuccess transitions) are shaded. Table 516 shows the nonsuccess transitions within the configuration process, and Table 517 shows further transition caused by events not belonging to the configuration process itself.

The configuration states in Table 515 through Table 517 and transitions are illustrated in Figure 215.

Table 515—CONFIG state/substates events: success transitions within configuration process

Previous state	Event	Condition	Action	Next state
WAIT_CONFIG	*ConfigureChannel_Req*	—	Send L2CAP_ ConfigReq	WAIT_CONFIG_ REQ_RSP
WAIT_CONFIG	L2CAP_ConfigReq	Options acceptable	Send L2CAP_ ConfigRsp (success)	WAIT_SEND _CONFIG
WAIT_CONFIG_ REQ_RSP	L2CAP_ConfigReq	Options acceptable	Send L2CAP_ ConfigRsp (success)	WAIT_CONFIG_ RSP
WAIT_CONFIG_ REQ_RSP	L2CAP_ConfigRsp	Remote side accepts options	— (continue waiting for configuration request)	WAIT_CONFIG_ REQ
WAIT_CONFIG_ REQ	L2CAP_ConfigReq	Options acceptable	Send L2CAP_ ConfigRsp (success)	OPEN
WAIT_SEND _CONFIG	*ConfigureChannel_Req*	—	Send L2CAP_ ConfigReq	WAIT_CONFIG_ RSP
WAIT_CONFIG_ RSP	L2CAP_ConfigRsp	Remote side accepts options	—	OPEN

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

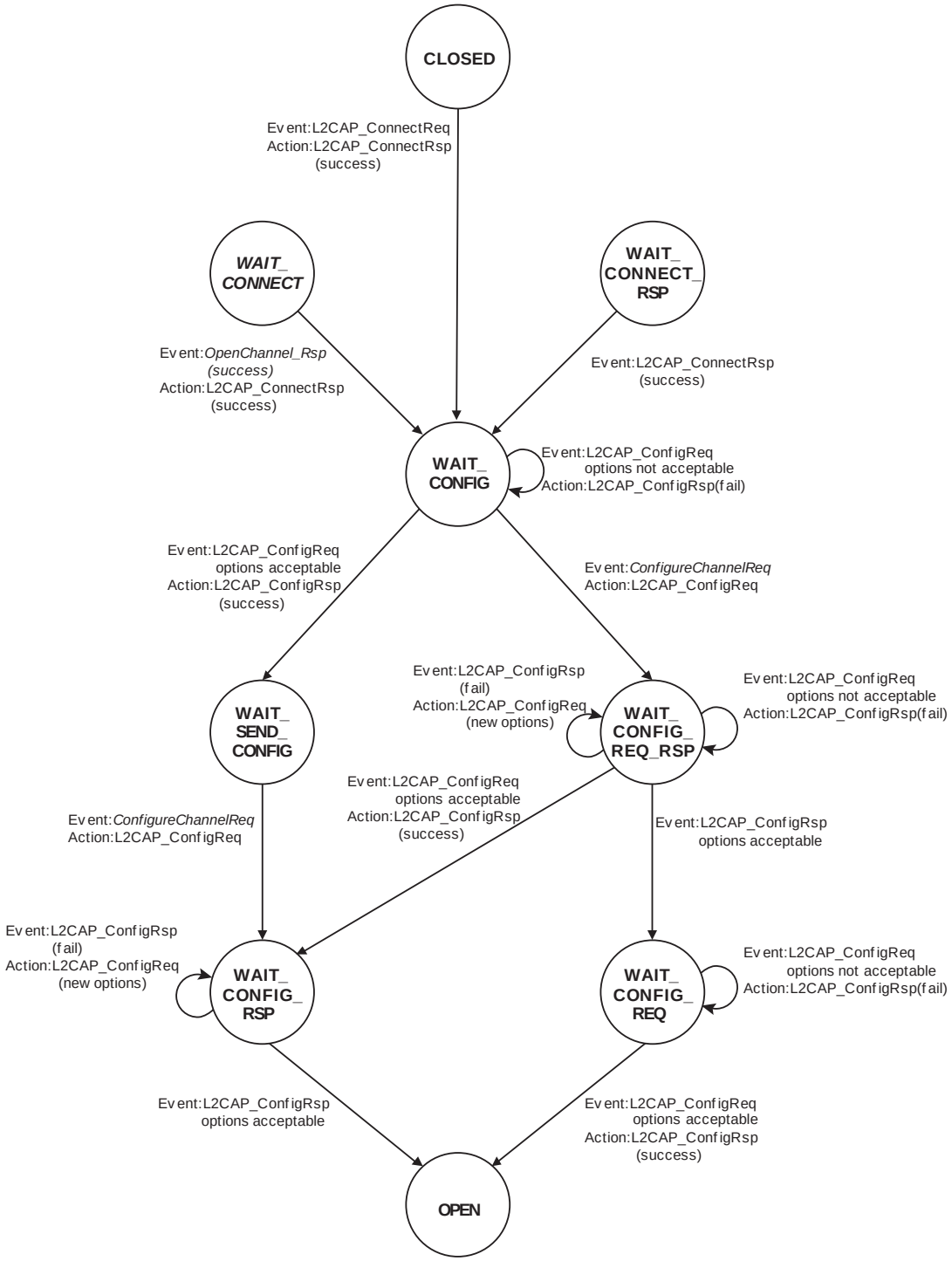


Figure 215—Configuration states and transitions

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 516—CONFIG state/substates events: nonsuccess transitions within configuration process

Previous state	Event	Condition	Action	Next state
WAIT_CONFIG	L2CAP_ConfigReq	Options not acceptable	Send L2CAP_ ConfigRsp (fail)	WAIT_CONFIG
WAIT_CONFIG	L2CAP_ConfigRsp	—	Ignore	WAIT_CONFIG
WAIT_SEND _CONFIG	L2CAP_ConfigRsp	—	Ignore	WAIT_SEND _CONFIG
WAIT_CONFIG_ REQ_RSP	L2CAP_ConfigReq	Options not acceptable	Send L2CAP_ ConfigRsp (fail)	WAIT_CONFIG_ REQ_RSP
WAIT_CONFIG_ REQ_RSP	L2CAP_ConfigRsp	Remote side rejects options	Send L2CAP_ ConfigReq (new options)	WAIT_CONFIG_ REQ_RSP
WAIT_CONFIG_ REQ	L2CAP_ConfigReq	Options not acceptable	Send L2CAP_ ConfigRsp (fail)	WAIT_CONFIG_ REQ
WAIT_CONFIG_ REQ	L2CAP_ConfigRsp	—	Ignore	WAIT_CONFIG_ REQ
WAIT_CONFIG_ RSP	L2CAP_ConfigRsp	Remote side rejects options	Send L2CAP_ ConfigReq (new options)	WAIT_CONFIG_ RSP

Table 517—CONFIG state/substates events: events not related to configuration process

Previous state	Event	Condition	Action	Next state
CONFIG (any substate)	*CloseChannel_Req*	Any internal reason to stop	Send L2CAP_ DisconnectReq	WAIT_ DISCONNECT
CONFIG (any substate)	L2CAP_DisconnectReq	—	Send L2CAP_ DisconnectRsp	CLOSED
CONFIG (any substate)	L2CAP_DisconnectRsp	—	Ignore	CONFIG (remain in substate)
CONFIG (any substate)	L2CAP_Data	—	Process the PDU	CONFIG (remain in substate)

NOTES

1—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 timeout.

2—Indicating a failure in a Configuration Response packet 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 packet may be triggered.

14.6.1.5 OPEN state

See Table 518 for the events associated with the OPEN state.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 518—OPEN state events

Event	Condition	Action	Next state
SendData_req	—	Send L2CAP_Data packet according to configured mode	OPEN
ReconfigureChannel_req	—	Complete outgoing SDU Send L2CAP_ConfigReq	CONFIG (substate WAIT_CONFIG_RSP)
CloseChannel_req	—	Send L2CAP_DisconnectReq	WAIT_DISCONNECT
L2CAP_ConnectRsp	—	Ignore	OPEN
L2CAP_ConfigReq	Incoming config. options acceptable	Complete outgoing SDU Send L2CAP_ConfigRsp (ok)	CONFIG (substate WAIT_CONFIG_REQ)
L2CAP_ConfigReq	Incoming config. options not acceptable	Complete outgoing SDU Send L2CAP_ConfigRsp (fail)	OPEN
L2CAP_DisconnectReq	—	Send L2CAP_DisconnectRsp	CLOSED
L2CAP_DisconnectRsp	—	Ignore	OPEN
L2CAP_Data	—	Process the PDU	OPEN

14.6.1.6 WAIT_DISCONNECT state

See Table 519 for the events associated with the WAIT_DISCONNECT state.

Table 519—WAIT_DISCONNECT state events

Event	Condition	Action	Next state
L2CAP_ConnectRsp	—	Ignore	WAIT_DISCONNECT
L2CAP_ConfigReq	—	Send L2CAP_CommandReject with reason code *invalid CID in request*	WAIT_DISCONNECT
L2CAP_ConfigRsp	—	Ignore	WAIT_DISCONNECT
L2CAP_DisconnectReq	—	Send L2CAP_DisconnectRsp	CLOSED
L2CAP_DisconnectRsp	—	—	CLOSED
L2CAP_Data	—	Ignore	WAIT_DISCONNECT

14.6.2 Timers events

14.6.2.1 Response timeout expired (RTX) timer

The RTX timer is used to terminate the channel when the remote endpoint is unresponsive to signalling requests. This timer is started when a signalling request (see 14.7) 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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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 that 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 signalling CID that should never be terminated. The decision should be based on the flush timeout of the signalling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level.

For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq message and to enter WAIT_ DISCONNECT state. Channels can be transitioned directly to the CLOSED state.

The value of this timer is implementation dependent. However, the minimum initial value is 1 s, and the maximum initial value is 60 s. One RTX timer shall exist for each outstanding signalling request, including each Echo Request packet. 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 s.

14.6.2.2 Extended response timeout expired (ERTX) timer

The 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 message with a result = 0x0001 (connect pending) 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 in 14.6.2.1.

The value of this timer is implementation dependent. However, the minimum initial value is 60 s, and the maximum initial value is 300 s. 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 s. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq message and enter WAIT_DISCONNECT state. Channels should be transitioned directly to the CLOSED state.

14.7 General procedures

This subclause describes the general operation of L2CAP, including the configuration process and the handling and processing of user data for transportation over the air interface. This subclause 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 14.8.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.7.1 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:

a) The local device informs the remote side of the nondefault parameters that the local side will accept using a Configuration Request packet.
b) Remote side responds, agreeing or disagreeing with these values, including the default ones, using a Configuration Response packet.
c) The local and remote devices repeat steps a and b 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 s.

14.7.1.1 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 520 defines the configuration options that may be placed in a Configuration Request packet.

Table 520—Parameters allowed in request

Parameter	Description
MTU	Incoming MTU information
FlushTO	Outgoing flush timeout
QoS	Outgoing QoS information
RFCMode	Incoming retransmission and flow control mode

The state machine for the configuration process is described in 14.6.

14.7.1.2 Response path

The response path can configure the following:

— Responder's outgoing MTU (the remote side's incoming MTU).
— Remote side's flush timeout.
— Responder's incoming QoS flow specification (remote side's outgoing QoS flow specification).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— 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 nondefault value by including the proposed value in a negative response message.

Table 521—Parameters allowed in response

Parameter	Description
MTU	Outgoing MTU information
FlushTO	Incoming flush timeout
QoS	Incoming QoS information
RFCMode	Outgoing retransmission and flow control mode

14.7.2 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.

14.7.2.1 Fragmentation of L2CAP PDUs

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

The two LLID bits defined in the first octet of BB 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 216. An example of how fragmentation might be used in a device with HCI is shown in Figure 217.

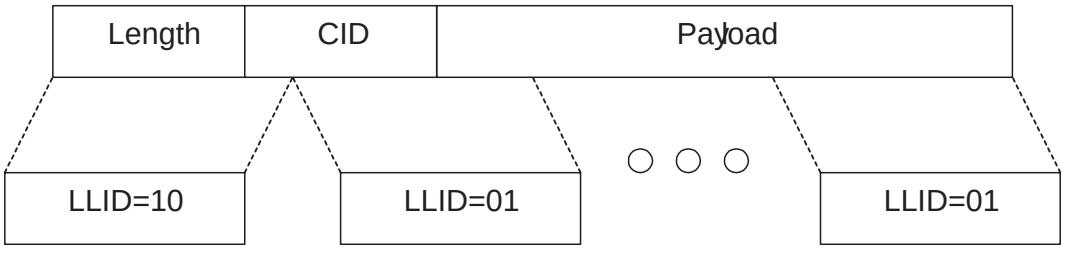


Figure 216—L2CAP fragmentation

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 BB packets. Thus, both L2CAP and the link controller use the same mechanism to control the size of fragments.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

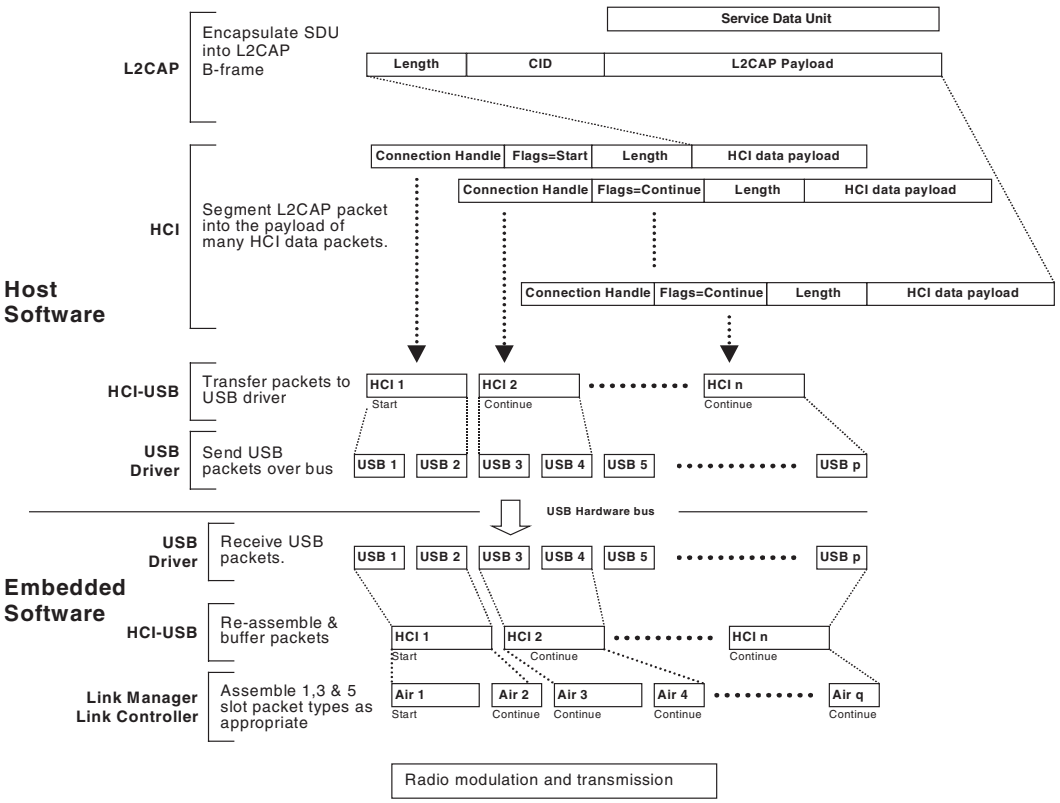


Figure 217—Example of fragmentation processes in a device with HCI

14.7.2.2 Recombination of L2CAP PDUs

The LCP 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 BB, it uses an 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 BB controller receives ACL packets, it either signals the L2CAP layer on the arrival of each BB 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 14.3) 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 14.5.2. For higher data integrity, L2CAP should be operated in the retransmission mode.

14.7.3 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 elements, resulting in a type of L2CAP PDU called a *basic information frame* (B-frame).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

SAR operations are used only in 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 elements resulting in an L2CAP PDU called an *information frame* (I-frame).

The maximum size of an SDU segment shall be given by the 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 bounderies shall be preserved between peer upper layer entities.

14.7.3.1 Segmentation of L2CAP SDUs

In flow control or retransmission modes, incoming SDUs may be broken down into segments, which shall then be individually encapsulated with L2CAP 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 218 illustrates segmentation and fragmentation.

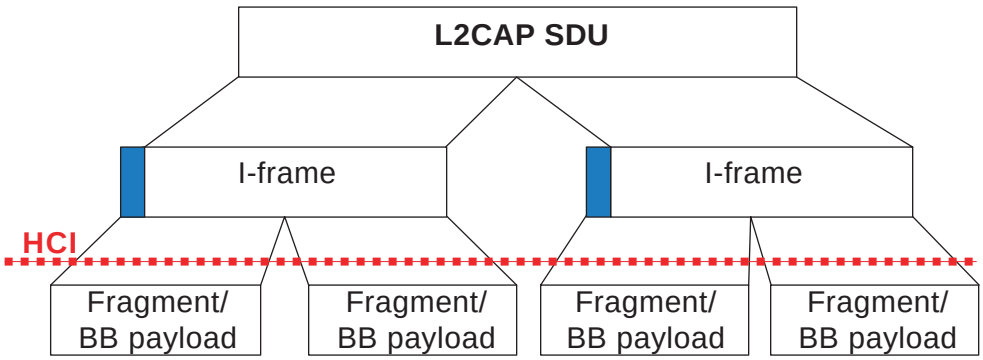


Figure 218—Segmentation and fragmentation of an SDU

14.7.3.2 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 SEQNs, may be used to indicate lost L2CAP SDUs to the application. Figure 218 illustrates segmentation and fragmentation.

14.7.3.3 Segmentation and fragmentation

Figure 219 illustrates the use of segmentation and fragmentation operations to transmit a single SDU.[18] 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 BB packets as created by the sender is common to both devices.

[18]For simplicity, the stripping of any additional HCI and universal serial bus (USB) specific information fields prior to the creation of the BB packets (e.g., Air_1, Air_2) is not shown in the figure.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

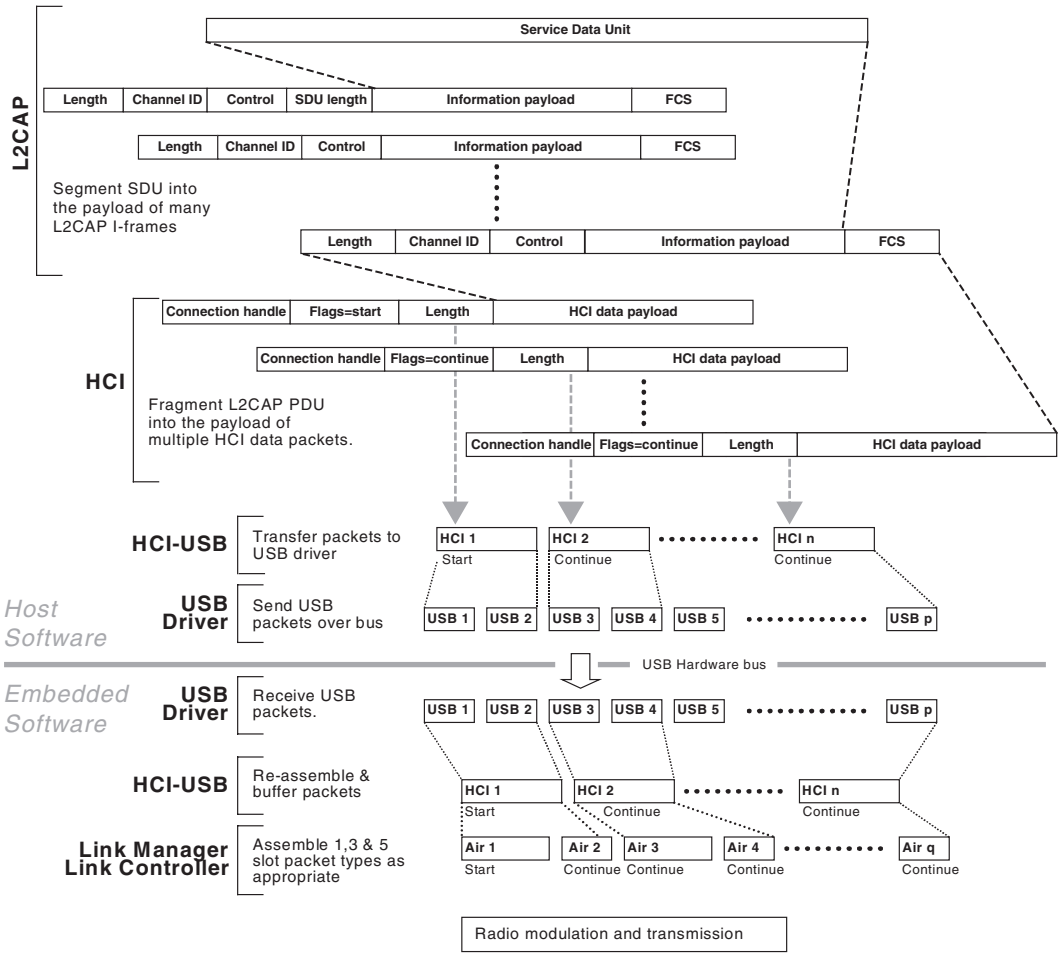


Figure 219—Example of segmentation and fragment processes in device with HCI

14.7.4 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.

14.7.5 Operation with flushing

In the L2CAP configuration, the flush timeout may be set separately per L2CAP channel, but there is only one flush mechanism per ACL logical transport in the BB.

When there is more than one L2CAP channel mapped to the same ACL logical transport, the automatic flush timeout does not discriminate between L2CAP channels. The automatic flush timeout flushes a specific L2CAP PDU. The HCI_Flush command flushes all outstanding L2CAP PDUs for the ACL logical transport. Therefore, care has to be taken when using the automatic flush timeout and the HCI_Flush command:

— For any connection to be reliable at the L2CAP level, it should use L2CAP retransmission mode if it is mapped to an ACL logical transport with a finite automatic flush timeout. In retransmission mode,

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

loss of flushed L2CAP PDUs on the channel is detected by the L2CAP ARQ mechanism, and they are retransmitted.

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

— If automatic flush timeout is used, then it should be taken into account that it flushes only one L2CAP PDU. If one PDU has timed out and needs flushing, then others on the same logical transport are also likely to need flushing. Therefore, when retransmission mode is used, flushing should be handled by the HCI_Flush command so that all outstanding L2CAP PDUs are flushed.

14.7.6 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 are sent in a best-effort manner. The connectionless channel has no QoS and is unreliable. L2CAP makes no guarantee that data sent through the connectionless channel successfully reach 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 these data are 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 transmisions on the connectionless channel; therefore, higher layer protocols must loop back 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 retransmission mode or flow control mode.

14.8 Procedures for flow control and retransmission

When flow control mode or retransmission mode is used, the procedures defined in this subclause shall be used, including the numbering of information frames, the handling of SDU SAR, and the detection and notification of errored frames. Retransmission mode also allows the sender to resend errored frames on request from the receiver.

14.8.1 Information retrieval

Before attempting to configure flow control or retransmission mode on a channel, it is mandatory to verify that the suggested mode is supported by performing an information retrieval when the information type = 0x0002 (Extended features supported). 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.

14.8.2 Function of PDU types for flow control and retransmission

Two frame formats are defined for flow control and retransmission modes (see 14.3.3). The I-frame is used to transport user information instead of the B-frame. The S-frame is used for signalling.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.8.2.1 I-frame

I-frames are sequentially numbered frames containing information fields. I-frames also include the functionality of RR S-frames (see 14.8.2.2.1).

14.8.2.2 S-frame

The S-frame is used to control the transmission of I-frames. The S-frame has two formats: receiver ready (RR) and reject (REJ).

14.8.2.2.1 RR S-frame

The RR S-frame is used to

— Acknowledge I-frames numbered up to and including ReqSeq – 1.

— Enable or disable retransmission of I-frames by updating the receiver with the current status of the Retransmission Disable bit.

The RR S-frame has no information field.

14.8.2.2.2 REJ S-frame

The 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 S-frame. The value of ReqSeq in the REJ S-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 S-frame is transmitted, it triggers a REJ Exception condition. A second REJ S-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 S-frame clears the REJ Exception condition. The REJ Exception condition applies only to traffic in one direction. Note that this means only valid I-frames can be rejected.

14.8.3 Variables and SEQNs

The sending peer uses the following variables and SEQNs:

— TxSeq – The send SEQN used to sequentially number each new I-frame transmitted.

— NextTxSeq – The SEQN to be used in the next new I-frame transmitted.

— ExpectedAckSeq – The SEQN of the next I-frame expected to be acknowledged by the receiving peer.

The receiving peer uses the following variables and SEQNs:

— ReqSeq – The SEQN sent in an acknowledgment frame to request transmission of I-frames 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 acknowledgment of a received I-frame so that new I-frame transmissions do not cause buffer overflow.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

All variables have the range of 0 to 63. Arithmetic operations on state variables (NextTXSeq, ExpectedTx-Seq, ExpectedAckSeq, BufferSeq) and SEQNs (TxSeq, ReqSeq) contained in this standard 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.

14.8.3.1 Sending peer

14.8.3.1.1 TxSeq

I-frames contain TxSeq, the send SEQN 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.

14.8.3.1.2 Send state variable NextTXSeq

The CID sent in the I-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 SEQN 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 (TX window). The value of TX window shall be in the range of 1 to 32.

14.8.3.1.3 Acknowledge state variable ExpectedAckSeq

The CID sent in the I-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. ExpectedAck-Seq is the SEQN 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, Expect-edAckSeq shall be initialized to 0.

NOTE—If the next acknowledgment 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 220 shows TxWindow = 5 and three frames awaiting transmission. The frame F7 may be transmitted when the frame F2 is acknowledged. When the frame F7 is transmitted, TxSeq is set to the value of NextTX-Seq. 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.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

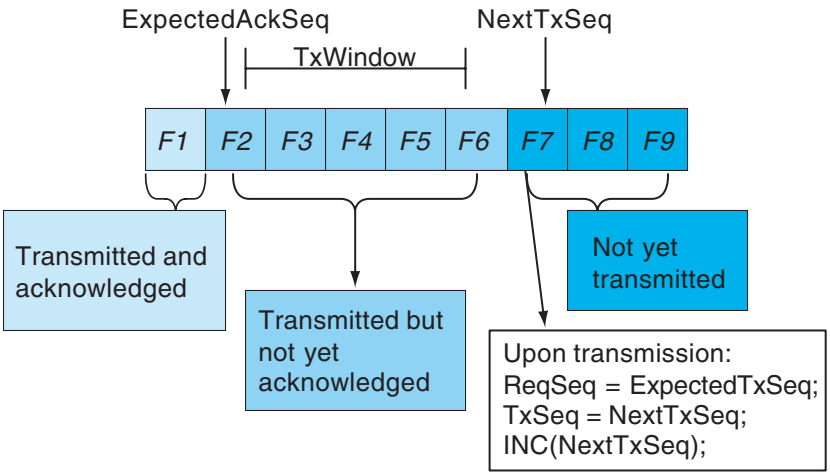


Figure 220—Example of transmitter side

14.8.3.2 Receiving peer

14.8.3.2.1 ReqSeq

All I-frames and S-frames contain ReqSeq, the send SEQN (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 DLL 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.

14.8.3.2.2 Receive state variable ExpectedTxSeq

Each channel shall have a receive state variable, ExpectedTxSeq. The receive state variable is the SEQN (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.

14.8.3.2.3 Buffer state variable BufferSeq

Each channel may have an associated buffer state variable, BufferSeq. BufferSeq is used to delay acknowledgment 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, e.g., 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 = 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.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

I-Frames shall have SEQNs in the range ExpectedTxSeq ≤ TxSeq < (BufferSeq + 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 221 shows TxWindow = 5. Frame F1 is successfully received and pulled by the upper layer. BufferSeq shows that frame 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 REJ Exception condition. The star indicates I-frames received, but discarded owing to the REJ Exception condition. They will be resent as part of the error recovery procedure.

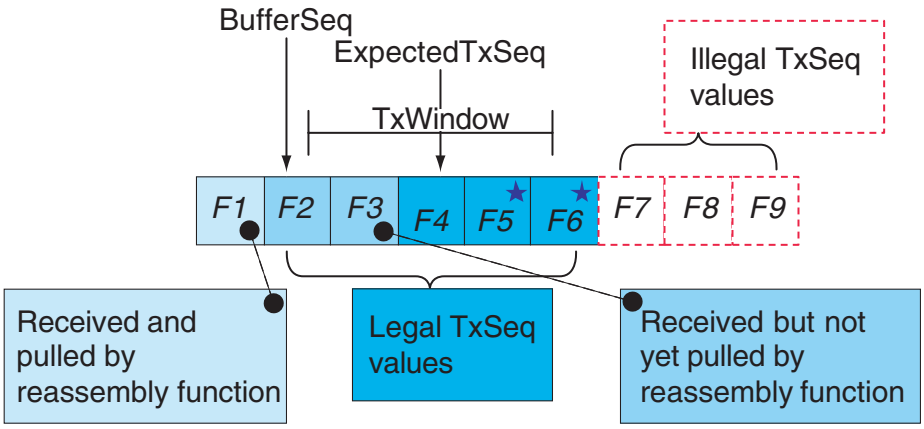


Figure 221—Example of receiver side

In Figure 221, there are several I-frames in a buffer awaiting the SDU reassembly function to pull them, and the TX window is full. The receiver would usually disable retransmission by setting the Retransmission Disable bit to 1 and send an RR S-frame 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.

14.8.4 Retransmission mode

14.8.4.1 Transmitting frames

A new I-frame shall be transmitted only when the TX window is not full. No I-frames shall be transmitted if the last Retransmission Disable (R) bit received is set to one.

A previously transmitted I-frame may be retransmitted as a result of an error recovery procedure, even if the TX window 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 R bit is stored and used along with the state of the retransmission timer to decide the actions when transmitting I-frames. The retransmission timer is running whenever I-frames have been sent, but not acknowledged.

14.8.4.1.1 Last received R bit set to zero

If the last R bit received was set to zero, then I-frames may be transmitted. If there are any I-frames that have been sent and not acknowledged, then they shall be retransmitted when the retransmission timer elapses. If

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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 retransmission timer has elapsed, then an unacknowledged I-frame shall be retransmitted. The retransmission timer shall be restarted.
b) If unacknowledged I-frames have been sent and 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 and the retransmission timer shall be started. If the monitor timer is running, it shall be stopped.
d) If no unacknowledged I-frames have been sent, if no new I-frames are waiting to be transmitted, and if the retransmission timer is running, then the retransmission timer shall be stopped and the monitor timer shall be started.

Table 522 summarizes actions when the retransmission timer is in use and R = 0.

Table 522—Summary of actions when the retransmission timer is in use and R = 0

Unacknowledged I-frames sent = Retransmission Timer is running	Retransmission Timer has elapsed	New I-frames are waiting	Transmit action	Timer action
True	True	True or False	Retransmit unacknowledged I-frame	Restart Retrans-mission 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 retransmis-sion timer
False	False	False	No Transmit action	If monitor timer is not running, then restart monitor timer

If the retransmission timer is not in use, no unacknowledged I-frames have been sent, and no new I-frames are waiting to be transmitted.

If the monitor timer is running and has not elapsed, then no transmit action shall be taken, and no timer action shall be taken.

If the monitor timer has elapsed, then an S-frame shall be sent, and the monitor timer shall be restarted.

If any I-frames become available for transmission, then the monitor timer shall be stopped, the retransmis-sion timer shall be started, and the rules for when the retransmission timer 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 Next-TxSeq shall be incremented by one.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.8.4.1.2 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 monitor timer is running and has not elapsed, then no transmit action shall be taken, and no timer action shall be taken.
b) If the monitor timer has elapsed, then an S-frame shall be sent, and the monitor timer shall be restarted.

14.8.4.2 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 14.8.4.6.

If a valid I-frame with TxSeq ≠ ExpectedTxSeq is received, then an exception condition shall be triggered which is handled according to 14.8.4.7.

14.8.4.3 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 acknowledgment 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 acknowledgment may either be an RR or an I-frame. The acknowledgment 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 monitor timer is active, then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.

14.8.4.4 Sending and receiving acknowledgments

Either the monitor timer or the retransmission timer shall be active while in retransmission mode. Both timers shall not be active concurrently.

14.8.4.4.1 Sending acknowledgments

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

14.8.4.4.2 Receiving acknowledgments

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.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The following rules shall be applied:

a) If the Retransmission Disable (R) bit changed value from 0 to 1 (stop retransmissions), then the receiving entity shall
   1) If the retransmission timer is running, then stop it and start the monitor timer.
   2) Store the state of the R bit received.
b) If the R bit changed value from 1 to 0 (start retransmissions), then the receiving entity shall
   1) Store the state of the R bit received.
   2) If there are any I-frames that have been sent, but not acknowledged, then stop the monitor timer and start the retransmission timer.
   3) Any buffered I-frames shall be transmitted according to 14.8.4.1.
c) If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame and the R bit equals 1 (retransmissions stopped), then the receiving entity shall
   1) Follow the rules in 14.8.4.1.
d) If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame and the R bit equals 0 (retransmissions started), then the receiving entity shall
   1) If the retransmission timer is running, then stop it.
   2) If any unacknowledged I-frames have been sent, then restart the retransmission timer.
   3) Follow the rules in 14.8.4.1.
   4) If the retransmission timer is not running and the monitor timer is not running, then start the monitor timer.

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.

14.8.4.5 Receiving REJ S-frames

Upon receipt of a valid REJ S-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 S-frame 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.

14.8.4.6 Waiting acknowledgments

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

a) If the transmit counter is less than the MaxTransmit field value, then
   1) Increment the transmit counter.
   2) Retransmit the last unacknowledged I-frame, according to 14.8.4.1.
b) If the transmit counter is equal to the MaxTransmit field value, 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.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

14.8.4.7 Exception conditions

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

14.8.4.7.1 TxSeq sequence error exception condition

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received that contains a TxSeq value that 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 Retransmission Disable (R) bit shall be processed according to 14.8.4.4. The information payload 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 R bit shall be processed according to 14.8.4.4.

  A REJ exception condition is triggered, and an REJ S-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.

14.8.4.7.2 ReqSeq sequence error exception condition

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

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

14.8.4.7.3 Timer recovery error exception condition

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

The L2CAP entity that transmitted an unacknowledged I-frame shall, on the expiry of the retransmission timer, take appropriate recovery action as defined in 14.8.4.6.

14.8.4.7.4 Invalid frame exception condition

Any frame received that is invalid (as defined in 14.3.3.6) shall be discarded, and no action shall be taken as a result of that frame.

14.8.5 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, the following rules apply:

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— REJ frames shall not be used in flow control mode.

— The Retransmission Disable bit 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 subclause. See Figure 222.

Assuming that the TX window 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 TX 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.

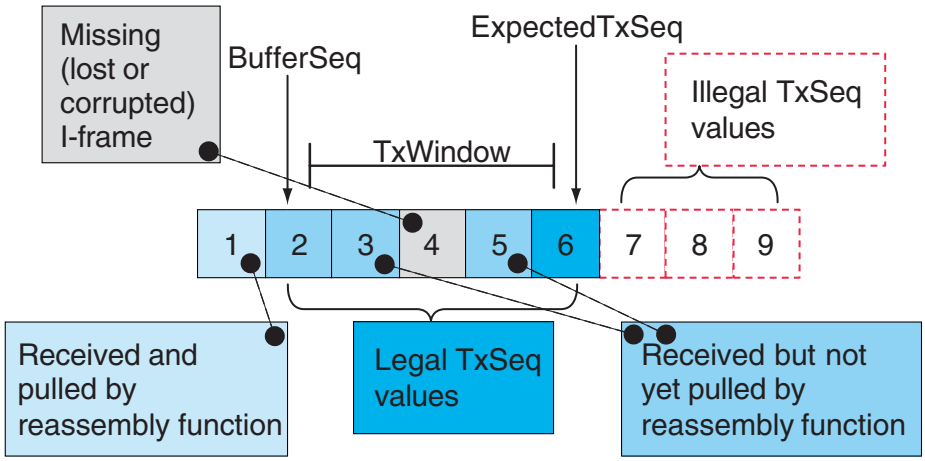


Figure 222—Overview of receiver side when operating in flow control mode

14.8.5.1 Transmitting I-frames

A new I-frame shall be transmitted only when the TX window 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 monitor timer shall be stopped, and the retransmission timer shall be started.

— If any I-frames have been sent and not acknowledged, then the retransmission timer 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.

14.8.5.2 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 acknowledgment 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 14.8.5.6), all frames with a SEQN 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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

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 acknowledgment shall not occur until the SDU reassembly function has pulled the I-frame. The ReqSeq shall be processed according to 14.8.5.4.

14.8.5.3 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 acknowledgment shall be sent to the peer entity. If the monitor timer 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 acknowledgment may be an RR S-frame or an I-frame. The acknowledgment 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.

14.8.5.4 Sending and receiving acknowledgments

One of the timers, monitor or retransmission, shall always be active while in flow control mode. Both timers shall never be active concurrently.

14.8.5.4.1 Sending acknowledgments

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

14.8.5.4.2 Receiving acknowledgments

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.

a) If any outstanding I-frames were acknowledged, then
    1) Stop the retransmission timer.
    2) If there are still unacknowledged I-frames, then restart the retransmission timer; otherwise, start the monitor timer.
    3) Transmit any I-frames awaiting transmission according to 14.8.5.1.

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

14.8.5.5 Waiting acknowledgments

If the retransmission timer expires, the following actions shall be taken:

a) The I-frame supervised by the retransmission timer shall be considered lost, and ExpectedAckSeq shall be incremented by one.
b) If I-frames are waiting to be sent,
    1) The retransmission timer is restarted.
    2) I-frames awaiting transmission are transmitted according to 14.8.5.1.
c) If no I-frames are waiting to be sent,

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

1) If there are still unacknowledged I-frames, the retransmission timer is restarted; otherwise, the monitor timer is started.

14.8.5.6 Exception conditions

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

14.8.5.6.1 TxSeq Sequence error exception condition

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received that contains a TxSeq value that 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 14.8.5.4. The information payload 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 of ExpectedTxSeq < TxSeq < (BufferSeq + TxWindow). The ReqSeq shall be processed according to 14.8.5.4.

  The missing I-frame(s) are considered lost, and ExpectedTXSeq is set equal to TxSeq+1 as specified in 14.8.5.2. 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.

14.8.5.6.2 ReqSeq Sequence error exception condition

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

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

14.8.5.6.3 Timer recovery error exception condition

An L2CAP entity that fails to receive an acknowledgment for an I-frame shall, on the expiry of the retransmission timer, take appropriate recovery action as defined in 14.8.5.5.

14.8.5.6.4 Invalid frame exception condition

Any frame received that is invalid (as defined in 14.3.3.6) 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.

14.9 Configuration MSCs

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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Figure 223 illustrates the basic configuration process. In this example, the devices exchange MTU information. All other values are assumed to be default.

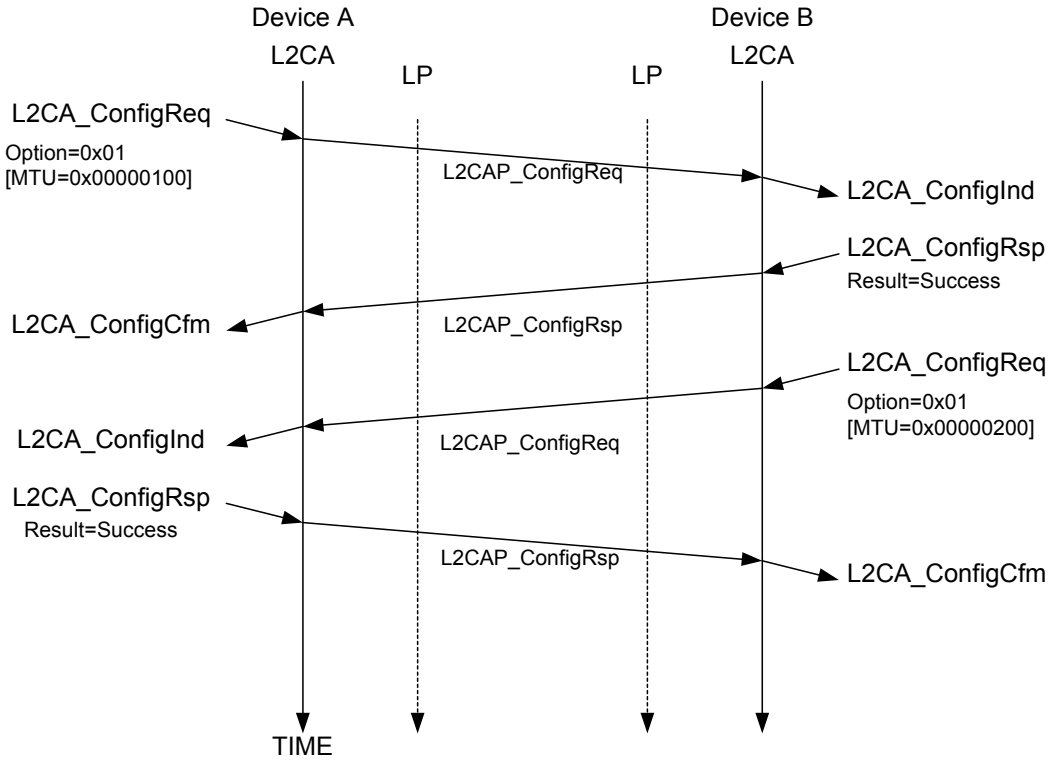


Figure 223—Basic MTU exchange

Figure 224 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 code *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 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 225 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).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

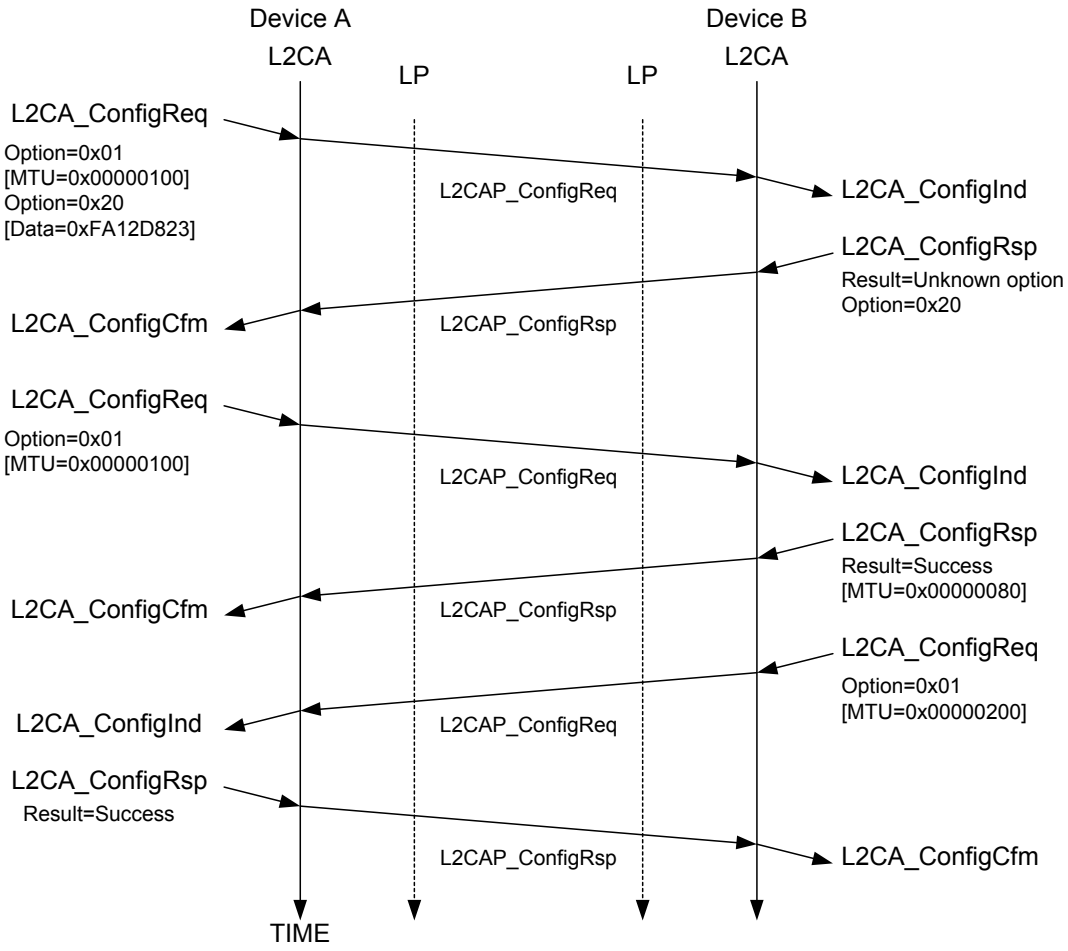


Figure 224—Dealing with unknown options

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

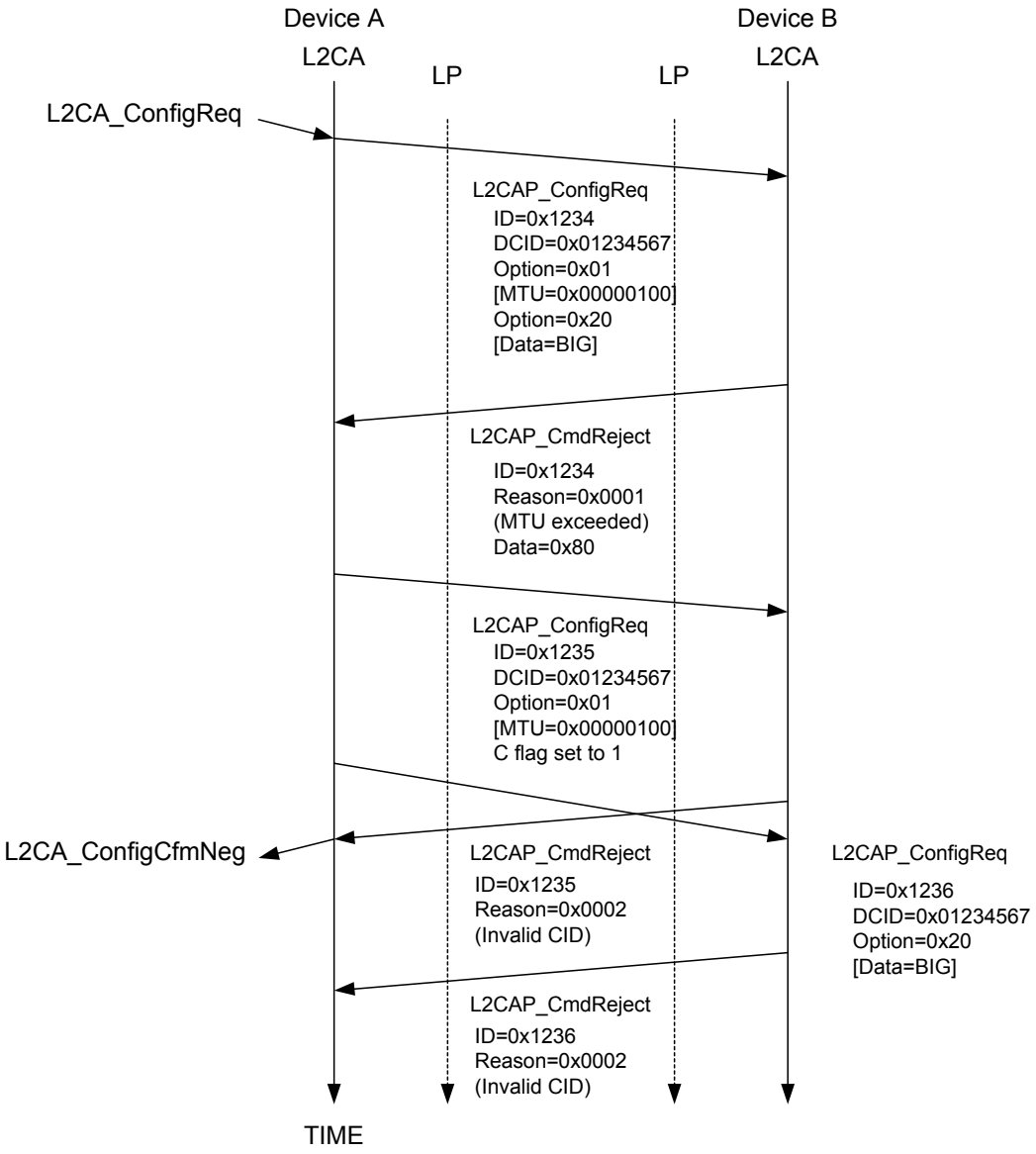


Figure 225—Unsuccessful configuration request

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15. Service access point (SAP) interfaces and primitives

15.1 IEEE 802® interfaces

Figure 226 indicates the relationship of the IEEE 802.15.1-2005 protocol stack to this clause. This clause describes the functions, features, protocol, services, and SAP interfaces between the MAC and logical link control (LLC) sublayers within the DLL of the ISO/IEC 8802 LAN Protocol (IEEE 802.2™). The LLC sublayer constitutes the top sublayer in the DLL (see Figure 227) and is common to the various medium access methods that are defined and supported by the ISO/IEC 8802 activity.

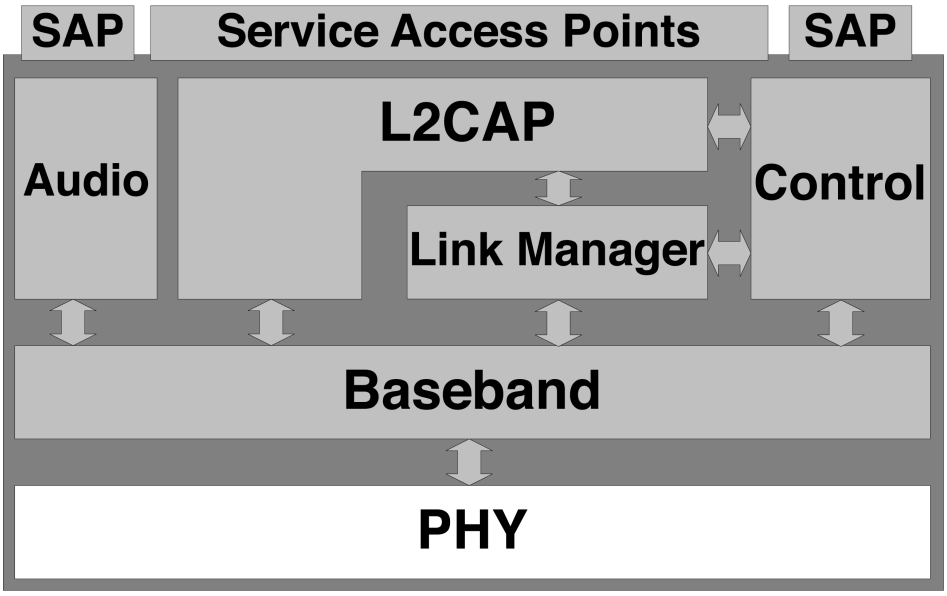


Figure 226—SAP relationships

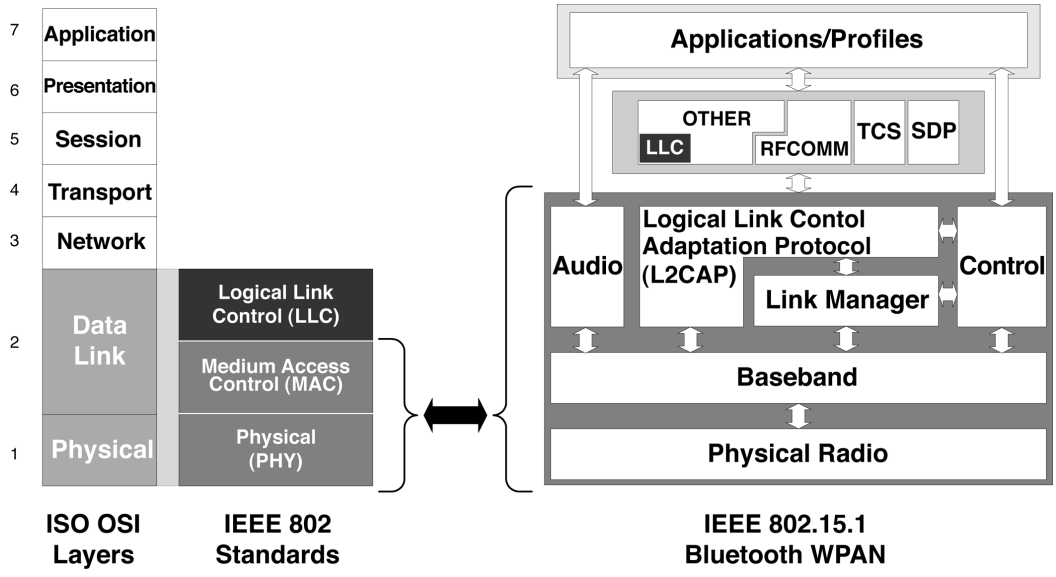


Figure 227—OSI and Bluetooth protocols

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The following concepts are discussed in this clause:

a) **DLL:** The conceptual layer of control or processing logic existing in the hierarchical structure of a station that is responsible for maintaining control of the data link. The DLL functions provide an interface between the station higher layer logic and the data link. These functions include address/control field interpretation, channel access and command, PDU/response, PDU generation, sending, and interpretation.

b) **LLC sublayer:** The part of a data station that supports the LLC functions of one or more logical links. The LLC generates command PDUs and response PDUs for sending and interprets received command PDUs and response PDUs. Specific responsibilities assigned to an LLC include the following:
   1) Initiation of control signal interchange
   2) Organization of data flow
   3) Interpretation of received command PDUs and generation of appropriate response PDUs
   4) Actions regarding error control and error recovery functions in the LLC sublayer

c) **MAC sublayer:** The part of a data station that supports the MAC functions that reside just below the LLC sublayer. The MAC procedures include framing/deframing data units, performing error checking, and acquiring the right to use the underlying physical medium.

   LLC sublayer service specifications to the MAC sublayer: The service specifications to the MAC sublayer provide a description of the services the LLC sublayer requires of the MAC sublayer. These services are defined to be independent of the form of the medium access methodology and the nature of the medium itself. All the above service specifications are given in the form of primitives that represent in an abstract way the logical exchange of information and control between the LLC sublayer and the identified service function (MAC sublayer). They do not specify or constrain the implementation of entities or interfaces.

d) **Type of operation:** One type of data link control (DLC) operation. Type 1 DLC operation provides a data-link-connectionless-mode service across a data link with minimum protocol complexity. This type of operation may be useful when higher layers provide any essential recovery and sequencing services so that these layers do not need replicating in the DLL. In addition, this type of operation may prove useful in applications where it is not essential to guarantee the delivery of every DLL data unit. This type of service is described in this clause in terms of logical data links.

   With Type 1 operation, PDUs shall be exchanged between LLCs without the need for the establishment of a data link connection. In the LLC sublayer, these PDUs shall not be acknowledged, and there shall not be any flow control or error recovery in Type 1 operation.

e) **Class of operation:** One class of LLC operation. Class I provides data-link-connectionless-mode service only.

   Class I LLCs shall support Type 1 operation only. Class I service shall be applicable to individual, group, global, and null addressing and applications requiring no DLL acknowledgment or flow control procedures.

   The basic protocols described here are peer protocols for use in multistation, multiaccess environments. Because of the multistation, multiaccess environment, it shall be possible for a station to be involved in a multiplicity of peer protocol data exchanges with a multiplicity of different stations over a multiplicity of different logical data links and/or data link connections that are carried by a single PHY over a single physical medium. Each unique toñfrom pairing at the DLL shall define a separate logical data link or data link connection with separate logical parameters and variables. Except where noted, the procedures described shall relate to each DLL logical data link or data link connection separately and independently of any other logical data link or data link connection that may exist at the stations involved.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.1.1 LLC sublayer service specifications (general)

This subclause covers the services required of, or by, the MAC sublayer. In general, the services of a layer (or sublayer) are the capabilities it offers a user in the next higher layer (or sublayer). To provide its service, a layer (or sublayer) builds its functions on the services it requires from the next lower layer (or sublayer). Figure 228 illustrates this notion of service hierarchy and shows the relationship of the two correspondent N users and their associated N layer (or sublayer) peer protocol entities.

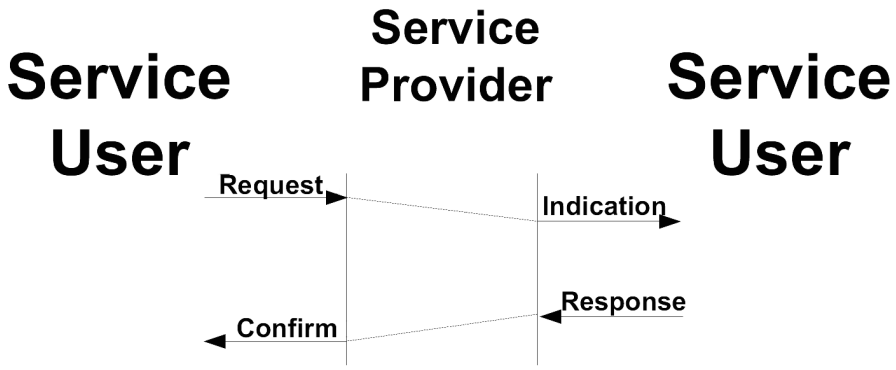


Figure 228—Service primitives

Services are specified by describing the information flow between the N user and the N layer (or sublayer). This information flow is modeled by discrete, instantaneous events that characterize the provision of a service. Each event consists of passing a service primitive from one layer (or sublayer) to the other through an N layer (or sublayer) SAP associated with an N user. Service primitives convey the information required in providing a particular service. These service primitives are an abstraction in that they specify only the service provided rather than the means by which that service is provided. This definition of service is independent of any particular interface implementation.

Services are specified by describing the service primitives and parameters that characterize each service. A service may have one or more related primitives that constitute the activity that is related to that service. Each service primitive may have zero or more parameters that convey the information required to provide that service.

Primitives are of four generic types as follows:

a) Request: The request primitive is passed from the N user to the N layer (or sublayer) to request that a service be initiated.
b) Indication: The indication primitive is passed from the N layer (or sublayer) to the N user to indicate an internal N layer (or sublayer) event that is significant to the N user. This event may be logically related to a remote service request or may be caused by an event internal to the N layer (or sublayer).
c) Response: The response primitive is passed from the N user to the N layer (or sublayer) to complete a procedure previously invoked by an indication primitive.
d) Confirm: The confirm primitive is passed from the N layer (or sublayer) to the N user to convey the results of one or more associated previous service request(s).

Possible relationships among primitive types are illustrated by the time-sequence diagrams shown in Figure 229. The figure also indicates the logical relationship of the primitive types. Primitive types that occur earlier in time and are connected by dotted lines in the diagrams are the logical antecedents of subsequent primitive types.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

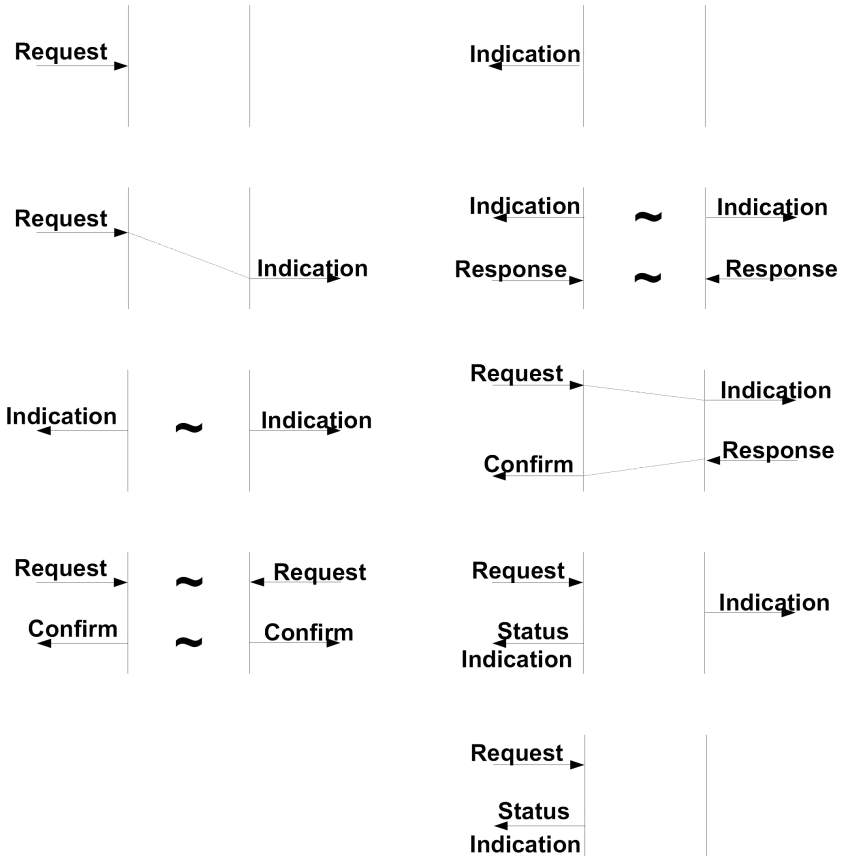


Figure 229—Sequence diagrams

15.2 LLC sublayer/MAC sublayer interface service specification

This subclause specifies the services required of the MAC sublayer by the LLC sublayer to allow the local LLC sublayer entity to exchange LLC data units with peer LLC sublayer entities. The services are described in an abstract way and do not imply any particular implementation or any exposed interface.

NOTE—Work is in progress to produce a single-service specification common to all the MAC sublayers. When this is available, it will be referenced in this standard instead of the current MAC service text.

The interactions described are as follows:

— MA-UNITDATA request

— MA-UNITDATA indication

— MA-UNITDATA-STATUS indication

15.2.1 MA-UNITDATA request

15.2.1.1 Function

This primitive requests the transfer of a MAC service data unit (MSDU) from a local LLC sublayer entity to a single peer LLC sublayer entity or multiple peer LLC sublayer entities in the case of group addresses.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.2.1.2 Semantics of the service primitive

The primitive shall provide parameters as follows:

```
MA-UNITDATA request (
                        source_address,
                        destination_address,
                        routing_information,
                        data,
                        priority,
                        service_class
                        )
```

The source_address parameter shall specify an individual MAC sublayer entity address. The destination_address parameter shall specify either an individual or a group MAC sublayer entity address. Together they shall contain sufficient information to create the source address (SA) and destination address (DA) fields that are appended to the LLC SDU by the local MAC sublayer entity as well as any physical layer address information (e.g., transmit frequency in broadband applications). The routing_information parameter specifies the route desired for the data unit transfer (a null value indicates that source routing is not to be used). The data parameter specifies the MSDU to be transmitted by the MAC sublayer entity, which includes the DSAP, SSAP, C, and information (if present) fields as specified in Section 3 of ISO/IEC 8802-2:1998(E), as well as sufficient information for the MAC sublayer entity to determine the length of the data unit. The priority parameter specifies the priority desired for the data unit transfer. The service_class parameter specifies the class of service desired for the data unit transfer.

15.2.1.3 When generated

This primitive is generated by the LLC sublayer entity to request that an MSDU be transferred to a peer LLC sublayer entity or entities. This can occur as a result of a request from higher layers of protocol or from an LSDU generated internally in the LLC sublayer, such as that required by Type 2 operation.

15.2.1.4 Effect on receipt

The receipt of this primitive shall cause the MAC sublayer entity to append all MAC-specified fields, including DA, SA, and any fields that are unique to the particular medium access method, and pass the properly formatted frame to the lower layers of protocol for transfer to the peer MAC sublayer entity or entities for subsequent transfer to the associated LLC sublayer(s).

15.2.1.5 Additional comments

A possible logical sequence of primitives associated with successful MSDU transfer is illustrated in Figure 229.

15.2.2 MA-UNITDATA indication

15.2.2.1 Function

This primitive defines the transfer of an MSDU from the MAC sublayer entity to the LLC sublayer entity or entities in the case of group addresses. In the absence of errors, the contents of the data parameter are logically complete and unchanged relative to the data parameter in the associated MA-UNITDATA request primitive.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.2.2.2 Semantics of the service primitive

The primitive shall provide parameters as follows:

```
MA-UNITDATA indication (
                        source_address,
                        destination_address,
                        routing_information,
                        data,
                        reception_status,
                        priority,
                        service_class
                        )
```

The source_address parameter shall specify an individual address as specified by the SA field of the incoming frame. The destination_address parameter shall be either an individual or a group address as specified by the DA field of the incoming frame. The routing_information parameter specifies the route used for the data unit transfer (for MAC sublayers that do not support source routing, this field will be set to null). The data parameter specifies the MSDU as received by the local MAC entity. The reception_status parameter indicates the success or failure of the incoming frame. The priority parameter specifies the priority desired for the data unit transfer. The service_class parameter specifies the class of service desired for the data unit transfer.

15.2.2.3 When generated

The MA-UNITDATA indication primitive is passed from the MAC sublayer entity to the LLC sublayer entity or entities to indicate the arrival of a frame at the local MAC sublayer entity. Frames are reported only if, at the MAC sublayer, they are validly formatted and received without error and their destination address designates the local MAC sublayer entity.

15.2.2.4 Effect on receipt

The effect on receipt of this primitive by the LLC sublayer is dependent on the validity and content of the frame.

15.2.2.5 Additional comments

If the local MAC sublayer entity is designated by the destination_address parameter of an MA-UNITDATA request primitive, the indication primitive will also be invoked by the MAC sublayer entity to the local LLC sublayer entity. This full-duplex characteristic of the MAC sublayer may be due to unique functionality within the MAC sublayer or full-duplex characteristics of the lower layers (e.g., all frames transmitted to the broadcast address will invoke MA-UNITDATA indication primitives at all stations in the network, including the station that generated the request).

15.2.3 MA-UNITDATA-STATUS indication

15.2.3.1 Function

This primitive has local significance and shall provide the LLC sublayer with status information for a previous associated MA-UNITDATA request primitive.

15.2.3.2 Semantics of the service primitive

The primitive shall provide parameters as follows:

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

MA-UNITDATA-STATUS indication (
source_address,
destination_address,
transmission_status,
provided_priority,
provided_service_class
)

The source_address parameter shall be an individual MAC sublayer entity address as specified in the associated MA-UNITDATA request primitive. The destination_address parameter shall be either an individual or a group MAC sublayer entity address as specified in the associated MA-UNITDATA request primitive. The transmission_status parameter is used to pass status information back to the local requesting LLC sublayer entity. The types of status that can be associated with this primitive are dependent on the particular implementation as well as the type of MAC sublayer that is used (e.g., "excessive collisions" may be a status returned by a CSMA/CD MAC sublayer entity). The provided_priority parameter specifies the priority that was used for the associated data unit transfer. The provided_service_class parameter specifies the class of service provided for the data unit transfer.

15.2.3.3 When generated

The MA-UNITDATA-STATUS indication primitive is passed from the MAC sublayer entity to the LLC sublayer to indicate the status of the service provided for a previous associated MA-UNITDATA request primitive.

15.2.3.4 Effect on receipt

The effect on receipt of this primitive by the LLC sublayer is dependent on the type of operation employed by the LLC sublayer entity.

15.2.3.5 Additional comments

It is assumed that sufficient information is available to the LLC sublayer entity to associate the status with the appropriate request.

15.3 Bluetooth interfaces

Figure 230 illustrates the events and actions performed by an implementation of the L2CAP layer. Client and server SAPs simply represent the initiator of the request and the acceptor of the request, respectively. An application-level client would both initiate and accept requests. The naming convention is as follows: The interface between two layers (vertical interface) uses the prefix of the lower layer offering the service to the higher layer, e.g., L2CA. The interface between two entities of the same layer (horizontal interface) uses the prefix of the protocol (adding a P to the layer identification), e.g., L2CAP. Events coming from above are called *requests* (Req), and the corresponding replies are called *confirms* (Cfm). Events coming from below are called *indications* (Ind), and the corresponding replies are called *responses* (Rsp). Responses that require further processing are called *pending* (Pnd). The notation for confirms and responses assumes positive replies. Negative replies are denoted by a "Neg" suffix such as L2CAP_ConnectCfmNeg.

While requests for an action always result in a corresponding confirmation (for the successful or unsuccessful satisfaction of the action), indications do not always result in corresponding responses. This is especially true if the indications are informative about locally triggered events, e.g., seeing the LP_QoSViolationInd or L2CA_TimeOutInd.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

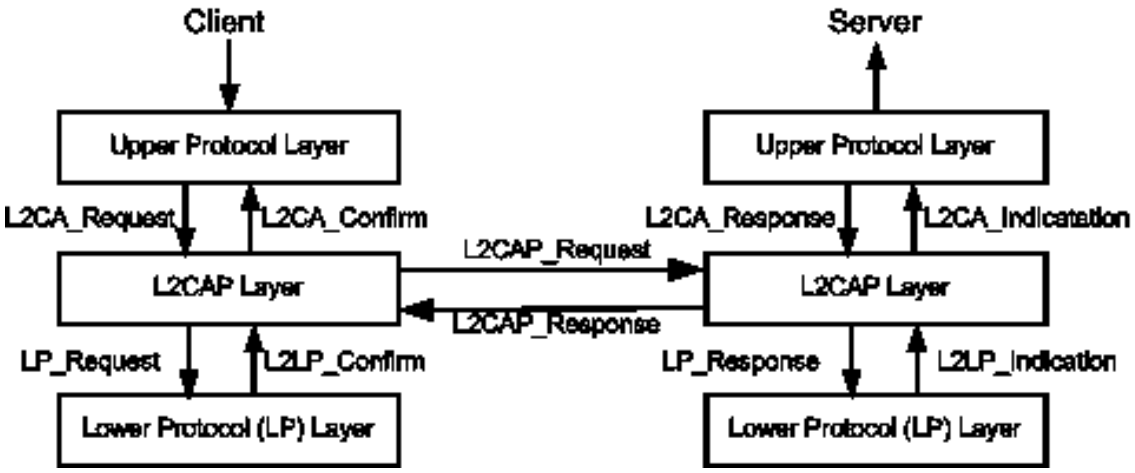


Figure 230—L2CAP actions and events

15.3.1 MSC of layer interactions

Figure 231 uses a MSC to illustrate the normal sequence of events. The two outer vertical lines represent the L2CA interface on the initiator (the device issuing a request) and the acceptor (the device responding to the initiator's request). Request commands at the L2CA interface result in requests defined by the protocol. When the protocol communicates the request to the acceptor, the remote L2CA entity presents the upper protocol with an indication. When the acceptor's upper protocol responds, the response is packaged by the protocol and communicated back the to initiator. The result is passed back to the initiator's upper protocol using a confirm message.

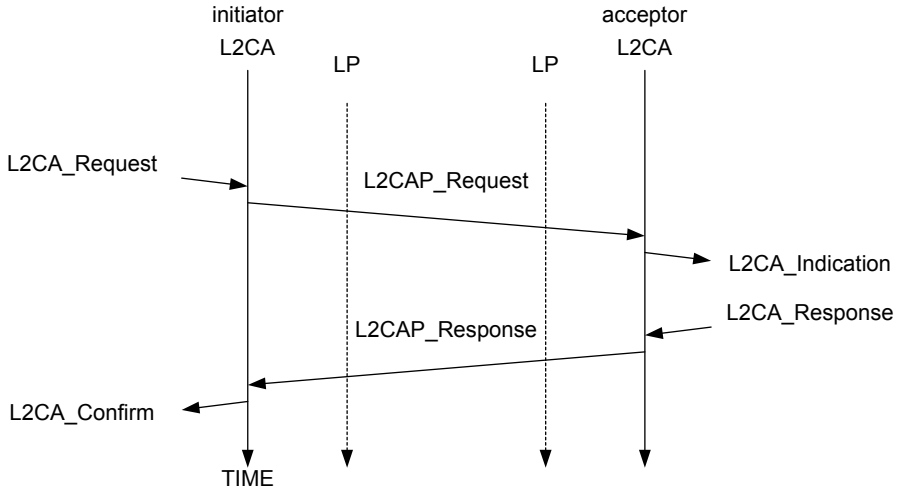


Figure 231—L2CA interaction

15.3.2 Relationship of Bluetooth protocol entities to IEEE 802 constructs

Figure 232 shows a mapping of the IEEE 802 protocol concepts to the Bluetooth components described in this standard. The PHY is all of the radio portion and part of the BB. The MAC contains the L2CAP and the rest of the BB. The LM, LMP, and HCI functions form the MAC management function. The PHY management functions, like synchonization and generation of various frequency hopping sequences generation, are incorporated within the BB.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

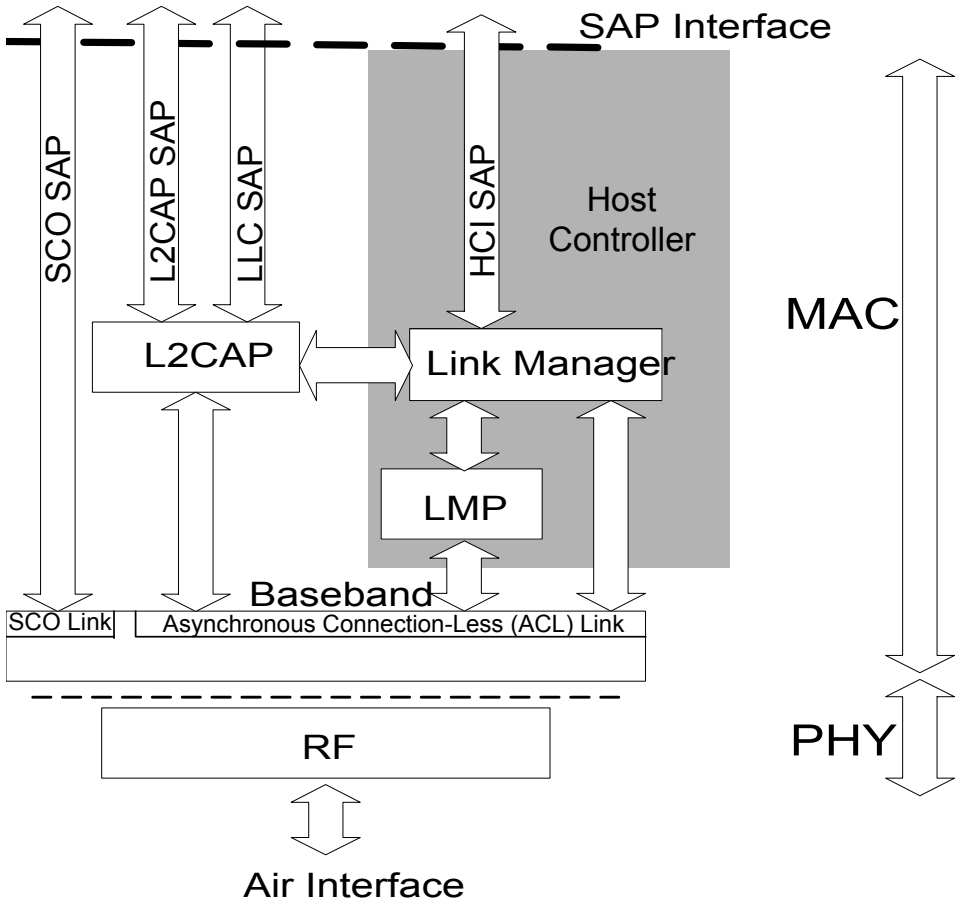


Figure 232—Bluetooth protocol entities mapped onto IEEE 802 constructs

15.3.2.1 Client SAP interface (upper layer) to L2CAP request event primitives

L2CA_ConnectReq (15.3.4.2)

L2CA_ConfigReq (15.3.4.4)

L2CA_DisconnectReq (15.3.4.6)

L2CA_DataWrite (15.3.4.7)

L2CA_DataRead (15.3.4.8)

L2CA_GroupCreate (15.3.4.9)

L2CA_GroupClose (15.3.4.10)

L2CA_GroupAddMember (15.3.4.11)

L2CA_GroupRemoveMember (15.3.4.12)

L2CA_GroupMembership (15.3.4.13)

L2CA_Ping (15.3.4.14)

L2CA_InfoReq (15.3.4.15)

L2CA_DisableCLT (15.3.4.16)

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

L2CA_EnableCLT (15.3.4.17)

15.3.2.2 Server SAP interface (upper layer) to L2CAP response events primitives

L2CA_ConnectRsp (15.3.4.3)

L2CA_ConnectRspNeg (15.3.4.3)

L2CA_ConnectPnd (15.3.4.2)

L2CA_ConfigRsp (15.3.4.5)

L2CA_ConfigRspNeg (15.3.4.5)

L2CA_DisconnectRsp (15.3.4.6)

L2CA_DataWriteRsp (15.3.4.7)

L2CA_DataReadRsp (15.3.4.8)

L2CA_GroupCloseRsp (15.3.4.10)

L2CA_GroupAddMemberRsp (15.3.4.11)

L2CA_GroupRemoveMemberRsp (15.3.4.12)

L2CA_GroupMembershipRsp (15.3.4.13)

L2CA_EchoRsp (15.3.4.14)

L2CA_InfoRsp (15.3.4.15)

L2CA_DisableCLTRsp (15.3.4.16)

L2CA_EnableCLTRsp (15.3.4.17)

15.3.2.3 L2CAP to client SAP interface (upper layer) action primitives

L2CA_ConnectCfm (15.3.4.3)

L2CA_ConnectCfmNeg (15.3.4.3)

L2CA_ConfigCfm (15.3.4.5)

L2CA_ConfigCfmNeg (15.3.4.5)

L2CA_DisconnectCfm (15.3.4.6)

15.3.2.4 L2CAP to server SAP interface (upper layer) event indication primitives

L2CA_ConnectInd (15.3.4.1.1)

L2CA_ConfigInd (15.3.4.1.2)

L2CA_DisconnectInd (15.3.4.1.3)

L2CA_QosViolationInd (15.3.4.1.4)

L2CA_TimeOutInd (15.3.4.2)

15.3.2.5 HCI SAP primitives

HCI_Inquiry (11.7.1.1)

HCI_Inquiry_Cancel (11.7.1.2)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

HCI_Periodic_Inquiry_Mode (11.7.1.3)

HCI_Exit_Periodic_Inquiry_Mode (11.7.1.4)

HCI_Create_Connection (11.7.1.5)

HCI_Disconnect (11.7.1.6)

HCI_Create_Connection_Cancel (11.7.1.7)

HCI_Add_SCO_Connection (11.8.5)

HCI_Accept_Connection_Request (11.7.1.8)

HCI_Reject_Connection_Request (11.7.1.9)

HCI_Link_Key_Request_Reply (11.7.1.10)

HCI_Link_Key_Request_Negative_Reply (11.7.1.11)

HCI_PIN_Code_Request_Reply (11.7.1.12)

HCI_PIN_Code_Request_Negative_Reply (11.7.1.13)

HCI_Change_Connection_Packet_Type (11.7.1.14)

HCI_Authentication_Requested (11.7.1.15)

HCI_Set_Connection_Encryption (11.7.1.16)

HCI_Change_Connection_Link_Key (11.7.1.17)

HCI_Master_Link_Key (11.7.1.18)

HCI_Remote_Name_Request (11.7.1.19)

HCI_Read_Remote_Supported_Features (11.7.1.21)

HCI_Read_Remote_Extended_Features (11.7.1.22)

HCI_Read_Remote_Version_Information (11.7.1.23)

HCI_Read_Clock_Offest (11.7.1.24)

HCI_Read_LMP_Handle (11.7.1.25)

HCI_Setup_Synchronous_Connection (11.7.1.26)

HCI_Accept_Synchronous_Connection_Request (11.7.1.27)

HCI_Reject_Synchronous_Connection_Request (11.7.1.28)

HCI_Hold_Mode (11.7.2.1)

HCI_Sniff_Mode (11.7.2.2)

HCI_Exit_Sniff_Mode (11.7.2.3)

HCI_Park_Mode (11.7.2.4)

HCI_Exit_Park_Mode (11.7.2.5)

HCI_Qos_Setup (11.7.2.6)

HCI_Role_Discovery (11.7.2.7)

HCI_Switch_Role (11.7.2.8)

HCI_Read_Link_Policy_Settings (11.7.2.9)

HCI_Write_Link_Policy_Settings (11.7.2.10)

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

HCI_Read_Default_Link_Policy_Settings (11.7.2.11)

HCI_Write_Default_Link_Policy_Settings (11.7.2.12)

HCI_Flow_Specification (11.7.2.13)

HCI_Set_Event_Mask (11.7.3.1)

HCI_Reset (11.7.3.2)

HCI_Set_Event_Filter (11.7.3.3)

HCI_Flush (11.7.3.4)

HCI_Read_PIN_Type (11.7.3.5)

HCI_Write_PIN_Type (11.7.3.6)

HCI_Create_New_Unit_Key (11.7.3.7)

HCI_Read_Stored_Link_Key (11.7.3.8)

HCI_Write_Stored_Link_Key (11.7.3.9)

HCI_Delete_Stored_Link_Key (11.7.3.10)

HCI_Write_Local_Name (11.7.3.11)

HCI_Read_Local_Name (11.7.3.12)

HCI_Read_Connection_Accept_Timeout (11.7.3.13)

HCI_Write_Connection_Accept_Timeout (11.7.3.14)

HCI_Read_Page_Timeout (11.7.3.15)

HCI_Write_Page_Timeout (11.7.3.16)

HCI_Read_Scan_Enable (11.7.3.17)

HCI_Write_Scan_Enable (11.7.3.18)

HCI_Read_Page_Scan_Activity (11.7.3.19)

HCI_Write_Page_Scan_Activity (11.7.3.20)

HCI_Read_Inquiry_Scan_Activity (11.7.3.21)

HCI_Write_Inquiry_Scan_Activity (11.7.3.22)

HCI_Read_Authentication_Enable (11.7.3.23)

HCI_Write_Authentication_Enable (11.7.3.24)

HCI_Read_Encryption_Mode (11.7.3.25)

HCI_Write_Encryption_Mode (11.7.3.26)

HCI_Read_Class_of_Device (11.7.3.27)

HCI_Write_Class_of_Device (11.7.3.28)

HCI_Read_Voice_Setting (11.7.3.29)

HCI_Write_Voice_Setting (11.7.3.30)

HCI_Read_Automatic_Flush_Timeout (11.7.3.31)

HCI_Write_Automatic_Flush_Timeout (11.7.3.32)

HCI_Read_Num_Broadcast_Retransmissions (11.7.3.33)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

HCI_Write_Num_Broadcast_Retransmissions (11.7.3.34)

HCI_Read_Hold_Mode_Activity (11.7.3.35)

HCI_Write_Hold_Mode_Activity (11.7.3.36)

HCI_Read_Transmit_Power_level (11.7.3.37)

HCI_Read_Synchronous_Flow_Control_Enable (11.7.3.38)

HCI_Write_Synchronous_Flow_Control_Enable (11.7.3.39)

HCI_Set_Host_Controller_To_Host_flow_Control (11.7.3.40)

HCI_Host_Buffer_Size (11.7.3.41)

HCI_Host_Number_Of_Completed_Packets (11.7.3.42)

HCI_Read_Link_Supervision_Timeout (11.7.3.43)

HCI_Write_Link_Supervision_Timeout (11.7.3.44)

HCI_Read_Number_Of_Supported_IAC (11.7.3.45)

HCI_Read_Current_IAC_LAP (11.7.3.46)

HCI_Write_Current_IAC_LAP (11.7.3.47)

HCI_Read_Page_Scan_Period_Mode (11.7.3.48)

HCI_Write_Page_Scan_Period_Mode (11.7.3.49)

Set_AFH_Host_Channel_Classification (11.7.3.50)

HCI_Read_Inquiry_Scan_Type (11.7.3.51)

HCI_Write_Inquiry_Scan_Type (11.7.3.52)

HCI_Read_Inquiry_Mode (11.7.3.53)

HCI_Write_Inquiry_Mode (11.7.3.54)

HCI_Read_Page_Scan_Mode (11.7.3.55)

HCI_Write_Page_Scan_Mode (11.7.3.56)

Read_AFH_Channel_Assessment_Mode (11.7.3.57)

Write_AFH_Channel_Assessment_Mode (11.7.3.58)

HCI_Read_Local_Version_Information (11.7.4.1)

HCI_Read_Local_Supported_Features (11.7.4.2)

HCI_Read_Local_Supported_Features (11.7.4.3)

HCI_Read_Local_Extended_Features (11.7.4.4)

HCI_Read_Buffer_Size (11.7.4.5)

HCI_Read_Country_Code (11.8.4)

HCI_Read_BD_ADDR (11.7.4.6)

15.3.2.6 SCO SAP primitives

SCO PDU Indications

SCO PDU Requests

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.3.2.7 Possible HCI events (11.3.1)

Inquiry Complete Event (11.7.7.1)

Inquiry Result Event (11.7.7.2)

Connection Complete Event (11.7.7.3)

Connection Request Event (11.7.7.4)

Disconnection Complete Event (11.7.7.5)

Authentication Complete Event (11.7.7.6)

Remote Name Request Complete Event (11.7.7.7)

Encryption Change Event (11.7.7.8)

Change Connection Link Key Complete Event (11.7.7.9)

Master Link Key Complete Event (11.7.7.10)

Read Remote Supported Features Complete Event (11.7.7.11)

Read Remote Version Information Complete Event (11.7.7.12)

QoS Setup Complete Event (11.7.7.13)

Command Complete Event (11.7.7.14)

Command Status Event (11.7.7.15)

Hardware Error Event (11.7.7.16)

Flush Occurred Event (11.7.7.17)

Role Change Event (11.7.7.18)

Number Of Completed Packets Event (11.7.7.19)

Mode Change Event (11.7.7.20)

Return_Link_Keys Event (11.7.7.21)

PIN Code Request Event (11.7.7.22)

Link Key Request Event (11.7.7.23)

Link Key Notification Event (11.7.7.24)

Loopback Command Event (11.7.7.25)

Data Buffer Overflow Event (11.7.7.26)

Max Slots Change Event (11.7.7.27)

Read Clock Offset Complete Event (11.7.7.28)

Connection Packet Type Changed Event (11.7.7.29)

QoS Violation Event (11.7.7.30)

Page Scan Mode Change Event (11.7.7.31)

HCI Flow Specification Complete (11.7.7.32)

Inquiry Result with RSSI (11.7.7.33)

Read Remote Extended Features Complete (11.7.7.34)

Synchronous Connection Complete (11.7.7.35)

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Synchronous Connection Changed (11.7.7.36)

Page Scan Mode Event (11.8.1)

15.3.3 Upper layer interface definitions

The defintions in 15.3.3.1 and 15.3.3.2 have been brought forward for IEEE Std 802.15.1-2002 and do not reflect the contents of the Bluetooth specification version 1.2.

15.3.3.1 Upper layer to L2CAP events

— *L2CA_ConnectReq.* Request from upper layer for the creation of a channel to a remote device.

— *L2CA_ConnectRsp.* Response from upper layer to the indication of a connection request from a remote device.

— *L2CA_ConnectRspNeg.* Negative response (rejection) from upper layer to the indication of a connection request from a remote device.

— *L2CA_ConfigReq.* Request from upper layer to (re)configure the channel.

— *L2CA_ConfigRsp.* Response from upper layer to the indication of a (re)configuration request.

— *L2CA_ConfigRspNeg.* A negative response from upper layer to the indication of a (re)configuration request.

— *L2CA_DisconnectReq.* Request from upper layer for the immediate disconnection of a channel.

— *L2CA_DisconnectRsp.* Response from upper layer to the indication of a disconnection request.

— *L2CA_DataRead.* Request from upper layer for the transfer of received data from L2CAP entity to upper layer.

— *L2CA_DataWrite.* Request from upper layer for the transfer of data from the upper layer to L2CAP entity for transmission over an open channel.

15.3.3.2 L2CAP to upper layer actions

— *L2CA_ConnectInd.* Indicates a Connection Request packet has been received from a remote device.

— *L2CA_ConnectCfm.* Confirms that a Connection Request packet has been accepted.

— *L2CA_ConnectCfmNeg.* Negative confirmation (failure) of a Connection Request packet.

— *L2CA_ConnectPnd.* Confirms that a Connection Response packet (pending) has been received from the remote device.

— *L2CA_ConfigInd.* Indicates a Configuration Request packet has been received from a remote device.

— *L2CA_ConfigCfm.* Confirms that a Configuration Request packet has been accepted.

— *L2CA_ConfigCfmNeg.* Negative confirmation (failure) of a Configuration Request packet.

— *L2CA_DisconnectInd.* Indicates a Disconnection Request packet has been received from a remote device or the remote device has been disconnected because it has failed to respond to a signalling request.

— *L2CA_DisconnectCfm.* Confirms that a Disconnect Request packet has been processed by the remote device following the receipt of a Disconnection Response packet from the remote device. An RTX timer expiration for an outstanding Disconnect Request packet can substitute for a Disconnect Response packet and result in this action. Upon receiving this event, the upper layer knows the L2CAP channel has been terminated. There is no corresponding negative confirm.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— *L2CA_TimeOutInd.* Indicates that a RTX or ERTX timer has expired. This indication will occur an implementation-dependent number of times before the L2CAP implementation will give up and send a L2CA_DisconnectInd message.

— *L2CA_QoSViolationInd.* Indicates that the QoS agreement has been violated.

15.3.4 Service primitives

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

15.3.4.1 Event Indication primitive

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

Table 523—Event Indication

Service	Input parameters	Output parameters
EventIndication	Event, Callback	Result

Table 524—Event indication input parameters

Parameter	Size	Type	Value	Description
Event	2 octets	u_int	0x00	Reserved
			0x01	L2CA_ConnectInd
			0x02	L2CA_ConfigInd
			0x03	L2CA_DisconnectInd
			0x04	L2CA_QoSViolationInd
			other	Reserved for future use
Callback	N/A	function	L2CA_ConnectInd	BD_ADDR, CID, PSM, Identifier
			L2CA_ConfigInd	CID, OutMTU, InFlow, InFlushTO
			L2CA_DisconnectInd	CID
			L2CA_QoSViolationInd	BD_ADDR

Table 525—Event Indication output parameter

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Event successfully registered.
			0x0001	Event registration failed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.3.4.1.1 L2CA_ConnectInd callback

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

15.3.4.1.2 L2CA_ConfigInd callback

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

15.3.4.1.3 L2CA_DisconnectInd callback

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

15.3.4.1.4 L2CA_QoSViolationInd callback

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

15.3.4.2 Connect primitive

This primitive initiates the sending of an L2CAP_ConnectReq message and blocks until a corresponding L2CA_ConnectCfm(Neg) or L2CA_TimeOutInd event is received.

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

Table 526—Connect

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

Table 527—Connect input parameters

Parameter	Size	Type	Value	Description
PSM	2 octets	u_int	0xXXXX	Target PSM provided for the connection.
BD_ADDR	6 octets	unit	0xXXXXXX XXXXXX	Unique address of target device.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 528—Connect output parameters

Parameter	Size	Type	Value	Description
LCID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel if result = 0x0000; otherwise, set to 0.
Result	2 octets	unit	0xXXXX XXXXXX XX	Unique address of target device.
			0x0000	Connection successful and the CID identifies the local endpoint. Ignore Status parameter.
			0x0001	Connection pending. Check Status parameter for more information.
			0x0002	Connection refused because no service for the PSM has been registered.
			0x0003	Connection refused because the security architecture on the remote side has denied the request.
			0xEEEE	Connection timeout occurred. This is a result of a timer expiration indication being included in the connection confirm message.
Status	2 octets	unit	0x0000	No further information.
			0x0001	Authentication pending.
			0x0002	Authorization pending.

15.3.4.3 Connect Response primitive

This primitive represents the L2CAP_ConnectRsp message.

The use of this primitive issues a response to a connection request event indication. Input parameters are the remote device's 48-bit address, identifier sent in the request, local CID (LCID), the response code, and the status attached to the response code. The output parameter is the result of the service request.

This primitive must be called no more than once after receiving the callback indication. This primitive returns once the local L2CAP entity has validated the request. A successful return does indicate the response has been sent over the air interface.

Table 529—Connect Response

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

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 530—Connect Response input parameters

Parameter	Size	Type	Value	Description
BD_ADDR	6 octets	unit	0xXXXX XXXXXX XX	Unique address of target device.
Identifier	1 octet	unit	0xXX	This value must match the value received in the L2CA_ConnectInd event described in 15.3.4.1.1.
LCID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.
Response	2 octets	u_int	0x0000	Connection successful.
			0x0001	Connection pending.
			0x0002	Connection refused – PSM not supported.
			0x0003	Connection refused – security block.
			0x0004	Connection refused – no resources available.
			0xXXXX	Other connection response code.
Status	2 octets	unit	0x0000	No further information available.
			0x0001	Authentication pending.
			0x0002	Authorization pending.
			0xXXXX	Other status code.

Table 531—Connect Response output parameter

Parameter	Size	Type	Value	Description
Result	2 octets	unit	0x0000	Response successfully sent.
			0x0001	Failure to match any outstanding connection request.

15.3.4.4 Configure primitive

This primitive initiates the sending of an L2CAP_ConfigReq message and blocks until a corresponding L2CA_ConfigCfm(Neg) or L2CA_TimeOutInd event is received.

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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 532—Configure

Service	Input parameters	Output parameters
L2CA_ConfigReq	CID, InMTU, OutFlow, OutFlushTO, LinkTO	Result, InMTU, OutFlow, OutFlushTO

Table 533—Configure input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	Local CID.
InMTU	2 octets	u_int	0xXXXX	MTU this channel can accept.
OutFlow	*x* octets	Flow	FlowSpec	QoS parameters dealing with the traffic characteristics of the outgoing data flow.
OutFlushTO	2 octets	u_int	0xXXXX	Number of milliseconds to wait before an L2CAP packet that cannot be acknowledged at the physical layer is dropped.
			0x0000	Request to use the existing flush timeout value if one exists; otherwise, the default value (0xFFFF) will be used.
			0x0001	Perform no retransmissions at the BB layer.
			0xFFFF	Perform retransmission at the BB layer until the link timeout terminates the channel.
LinkTO	2 octets	u_int	0xXXXX	Number of milliseconds to wait before terminating an unresponsive link.

Table 534—Configure output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Configuration is successful. Parameters contain agreed-upon values.
			0x0001	Failure – invalid CID.
			0x0002	Failure – unacceptable parameters.
			0x0003	Failure – signalling MTU exceeded.
			0x0004	Failure – unknown options.
			0xEEEE	Configuration timeout occurred. This is a result of a timer expiration indication being included in the configuration confirm.
InMTU			0xXXXX	MTU that the remote unit will send across this channel (maybe less or equal to the InMTU input parameter).

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 534—Configure output parameters *(continued)*

Parameter	Size	Type	Value	Description
OutFlow			FlowSpec	QoS parameters dealing with the traffic characteristics of the agreed-upon outgoing data flow if Result parameter is successful. Otherwise, this represents the requested QoS.
OutFlushTO			0xXXXX	Number of milliseconds before an L2CAP packet that cannot be acknowledged at the physical layer is dropped. This value is informative of the actual value that will be used for outgoing packets. It may be less or equal to the OutFlushTO parameter given as input.

15.3.4.5 Configuration Response primitive

This primitive represents the L2CAP_ConfigRsp.

The use of this primitive issues a response to a configuration request event indication. Input parameters include the local CID of the endpoint being configured, outgoing transmit MTU (which may be equal or less to the OutMTU parameter in the L2CA_ConfigInd event), and the accepted flow specification for incoming traffic. The output parameter is the Result value.

Table 535—Configuration Response

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

Table 536—Configuration Response input parameters

Parameter	Size	Type	Value	Description
LCID	2 octets	u_int	0xXXXX	Local CID.
OutMTU	2 octets	u_int	0xXXXX	MTU this channel will send.
InFlow	*x* octets	Flow	FlowSpec	QoS parameters dealing with the traffic characteristics of the incoming data flow.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 537—Configuration Response output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Configuration is successful. Parameters contain agreed upon values.
			0x0001	Configuration failed – unacceptable parameters.
			0x0002	Configuration failed – rejected.
			0x0003	Configuration failed – invalid CID.
			0x0004	Configuration failed – unknown options.
			0xXXXX	Reserved.

15.3.4.6 Disconnect primitive

This primitive represents the L2CAP_DisconnectReq, and the returned output parameters represent the corresponding L2CAP_DisconnectRsp or the RTX timer expiration.

The use of this primitive requests the disconnection of the channel. Input parameter is the CID representing the local channel endpoint. Output parameter is the Result value. Result is zero if a L2CAP_DisconnectRsp is received; otherwise, a nonzero value is returned. Once disconnection has been requested, no process will be able to successfully read or write from the CID. Writes in progress should continue to be processed.

Table 538—Disconnect

Service	Input parameters	Output parameters
L2CA_DisconnectReq	CID	Result

Table 539—Disconnect input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.

Table 540—Disconnect output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Disconnection successful. This is a result of the receipt of a disconnection response message.
			0xEEEE	Disconnection timeout occurred.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.3.4.7 Write primitive

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

Table 541—Write

Service	Input parameters	Output parameters
L2CA_DataWrite	CID, Length, OutBuffer	Size, Result

Table 542—Write input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.
Length	2 octets	u_int	0xXXXX	Size, in bytes, of the buffer where data to be transmitted are stored.
OutBuffer	N/A	pointer	N/A	Address of the input buffer used to store the message.

Table 543—Write output parameters

Parameter	Size	Type	Value	Description
Size	2 octets	u_int	0xXXXX	The number of bytes transferred.
Result	2 octets	u_int	0x0000	Successful write.
			0x0001	Error – flush timeout expired.
			0x0002	Error – link termination (perhaps this should be left to the indication).

15.3.4.8 Read primitive

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

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 544—Read

Service	Input parameters	Output parameters
L2CA_DataRead	CID, Length, InBuffer	Result, N

Table 545—Read input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID.
Length	2 octets	u_int	0xXXXX	Size, in bytes, of the buffer where received data are to be stored.
InBuffer	N/A	pointer	N/A	Address of the buffer used to store the message.

Table 546—Read output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Success.
			0x0001	Unsuccessful.
N	2 octets	u_int	0xXXXX	Number of bytes transferred to InBuffer.

15.3.4.9 Group Create primitive

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

Table 547—Group Create

Service	Input parameters	Output parameters
L2CA_GroupCreate	PSM	CID

Table 548—Group Create input parameters

Parameter	Size	Type	Value	Description
PSM	2 octets	u_int	0xXXXX	Protocol/service multiplexer value.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 549—Group Create output parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.

15.3.4.10 Group Close primitive

The use of this primitive closes down a group.

Table 550—Group Close

Service	Input parameters	Output parameters
L2CA_GroupClose	CID	Result

Table 551—Group Close input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.

Table 552—Group Close output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Successful closure of the channel.
			0x0001	Invalid CID.

15.3.4.11 Group Add Member primitive

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

Table 553—Group Add Member

Service	Input parameters	Output parameters
L2CA_GroupAddMember	CID, BD_ADDR	Result

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 554—Group Add Member input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.
BD_ADDR	6 octets	u_int	0xXXXXXX XXXXXX	Remote device address.

Table 555—Group Add Member output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Success.
			0x0001	Failure to establish connection to remote device.
			Other	Reserved.

15.3.4.12 Group Remove Member primitive

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

Table 556—Group Remove Member

Service	Input parameters	Output parameters
L2CA_GroupRemoveMember	CID, BD_ADDR	Result

Table 557—Group Remove Member input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.
BD_ADDR	6 octets	u_int	0xXXXX XXXXXX XX	Unique address device to be removed.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 558—Group Remove Member output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Success.
			0x0001	Failure – device not a member of the group.
			Other	Reserved.

15.3.4.13 Get Group Membership primitive

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

Table 559—Get Group Membership

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

Table 560—Get Group Membership input parameters

Parameter	Size	Type	Value	Description
CID	2 octets	u_int	0xXXXX	CID representing local endpoint of the communication channel.

Table 561—Get Group Membership output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Success.
			0x0001	Failure – group does not exist.
			Other	Reserved.
N	2 octets	u_int	0x0000-0xFFFF	The number of devices in the group identified by the channel endpoint CID. If the Result parameter indicates failure, N should be set to 0.
BD_ADDR_List	n/a	pointer	0xXXXXXXXXXXXX	List of N unique addresses of the devices in the group identified by the channel endpoint CID. If the Result parameter indicates failure, the all-zero address is the only address that should be returned.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

15.3.4.14 Ping primitive

This primitive represents the initiation of an L2CA_EchoReq message and the reception of the corresponding L2CAP_EchoRsp message.

Table 562—Ping

Service	Input parameters	Output parameters
L2CA_Ping	BD_ADDR, ECHO_DATA, Length	Result, ECHO_DATA, Size

Table 563—Ping input parameters

Parameter	Size	Type	Value	Description
BD_ADDR	6 octets	u_int	0xXXXX XXXXXX XX	Unique address of target device.
ECHO_DATA	n/a	pointer	N/A	The buffer containing the contents to be transmitted in the data payload of the Echo Request command.
Length	2 octets	u_int	0xXXXX	Size, in bytes, of the data in the buffer.

Table 564—Ping output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Response received.
			0x0001	Timeout occurred.
ECHO_DATA	n/a	pointer	N/A	The buffer containing the contents received in the data payload of the Echo Response command.
Size	2 octets	u_int	0xXXXX	Size, in bytes, of the data in the buffer.

15.3.4.15 GetInfo primitive

This primitive represents the initiation of an L2CA_InfoReq message and the reception of the corresponding L2CAP_InfoRsp message.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 565—GetInfo

Service	Input parameters	Output parameters
L2CA_GetInfo	BD_ADDR, InfoType	Result, InfoData, Size

Table 566—GetInfo input parameters

Parameter	Size	Type	Value	Description
BD_ADDR	6 octets	unit	0xXXXXXX XXXXXX	Unique address of target device
InfoType	2 octets	unit	0x0001	Maximum MTU_{cnl} size

Table 567—GetInfo output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Response received.
			0x0001	Not supported.
InfoData		pointer		The buffer containing the contents received in the data payload of the Information Response command.
Size	2 octets	u_int	0xXXXX	Size, in bytes, of the data in the InfoData buffer.

15.3.4.16 Disable Connectionless Traffic primitive

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

Table 568—Disable Connectionless Traffic

Service	Input parameters	Output parameters
L2CA_DisableCLT	PSM	Result

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table 569—Disable Connectionless Traffic input parameters

Parameter	Size	Type	Value	Description
PSM	2 octets	u_int	0x0000	Block all connectionless traffic.
			0xXXXX	PSM field to be blocked.

Table 570—Disable Connectionless Traffic output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Successful.
			0x0001	Failure – not supported.

15.3.4.17 Enable Connectionless Traffic primitive

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

Table 571—Enable Connectionless Traffic

Service	Input parameters	Output parameters
L2CA_EnableCLT	PSM	Result

Table 572—Enable Connectionless Traffic input parameters

Parameter	Size	Type	Value	Description
PSM	2 octets	u_int	0x0000	Enable all connectionless traffic.
			0xXXXX	PSM field to enable.

Table 573—Enable Connectionless Traffic output parameters

Parameter	Size	Type	Value	Description
Result	2 octets	u_int	0x0000	Successful.
			0x0001	Failure – not supported.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Annex A

(informative)

Bibliography

The specifications refer to provisions that do not constitute provisions of this standard. They are listed to provide additional useful information. At the time of publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the references listed below.

[B1] "Bluetooth Assigned Numbers," Bluetooth Special Interest Group, [https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers].[19]

[B2] "Bluetooth Core Specification v1.2," Bluetooth Special Interest Group, [https://www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2].

[B3] "Bluetooth CVSD encoded test signal Version 2.1," A test signal file providing the digital CVSD encoded test signal as referred to in the Bluetooth Specification, Bluetooth Special Interest Group, 22 February 2001, [BluetoothSpecFiles_new_tar.gz].

[B4] "Bluetooth Network Encapsulation Protocol (BNEP) Specification Revision 0.95a," Bluetooth Special Interest Group, 12 June 2001, [BNEP.pdf].

[B5] "Bluetooth Personal Area Networking Profile Revision 0.95a," Bluetooth Special Interest Group, 26 June 2001, [PAN-Profile.pdf].

[B6] ETSI GSM 03.50, Transmission planning aspects of the speech service in the GSM PLMN system (Phase 1).[20]

[B7] IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition.[21]

[B8] "IEEE Style Guide," [http://standards.ieee.org/guides/style].

[B9] ITU-T Recommendation G.712, Transmission performance characteristics of pulse code modulation channels.[22]

[B10] ITU-T Recommendation G.714, Separate performance characteristics for the encoding and decoding sides of PCM channels applicable to 4-wire voice-frequency interfaces.

[B11] ITU-T Recommendation P.34, Transmission characteristics of hands-free telephones.

[B12] ITU-T Recommendation P.51, Telephone transmission quality—objective measuring apparatus—artificial mouth.

[19]Bluetooth documents are available from http://www.bluetooth.com/.

[20]ETSI publications are available from the European Telecommunications Standards Institute (http://www.etsi.org/).

[21]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

[22]ITU-T publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (http://www.itu.int/).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

[B13] ITU-T Recommendation P.57, Series P: Telephone transmission quality, telephone installations, local line networks—objective measuring apparatus—artificial ears.

[B14] ITU-T Recommendation P.79, Calculation of loudness ratings for telephone sets.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Annex B

(informative)

Generic access profile (GAP)

This informative annex defines the generic procedures related to discovery of devices (idle mode procedures) and link management aspects of connecting to devices (connecting mode procedures). It also defines procedures related to use of different security levels. In addition, this profile includes common format requirements for parameters accessible on the user interface level.

Interoperability between devices from different manufacturers is provided for a specific service and use case if the devices conform to a Bluetooth SIG-defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives a description of the air interface for specified service(s) and use case(s).

All defined features are process-mandatory. This means that, if a feature is used, it is used in a specified manner. Whether the provision of a feature is mandatory or optional is stated separately for both sides of the air interface.

B.1 Scope

The purpose of the GAP is as follows:

— To introduce definitions, recommendations, and common requirements related to modes and access procedures that are to be used by transport and application profiles.
— To describe how devices are to behave in standby and connecting states in order to guarantee that links and channels always can be established between devices and that multiprofile operation is possible. Special focus is put on discovery, link establishment, and security procedures.
— To state requirements on user interface aspects, mainly coding schemes and names of procedures and parameters, that are needed to guarantee a satisfactory user experience.

B.2 Symbols and conventions

B.2.1 Requirement status symbols

In this document (especially in the profile requirements tables), the following symbols are used:

— M for mandatory to support (used for capabilities that shall be used in the profile)
— O for optional to support (used for capabilities that can be used in the profile)
— C for conditional support (used for capabilities that shall be used in case a certain other capability is supported)
— X for excluded (used for capabilities that may be supported by the unit, but shall never be used in the profile)
— N/A for not applicable (in the given context it is impossible to use this capability)

Some excluded capabilities are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices following this profile. Therefore, these features shall never be activated while a unit is operating as a unit within this profile.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

In this specification, the word *shall* is used for mandatory requirements, the word *should* is used to express recommendations, and the word *may* is used for options.

B.2.2 Signaling diagram conventions

The arrows shown in Figure B.1 are used in diagrams describing procedures.

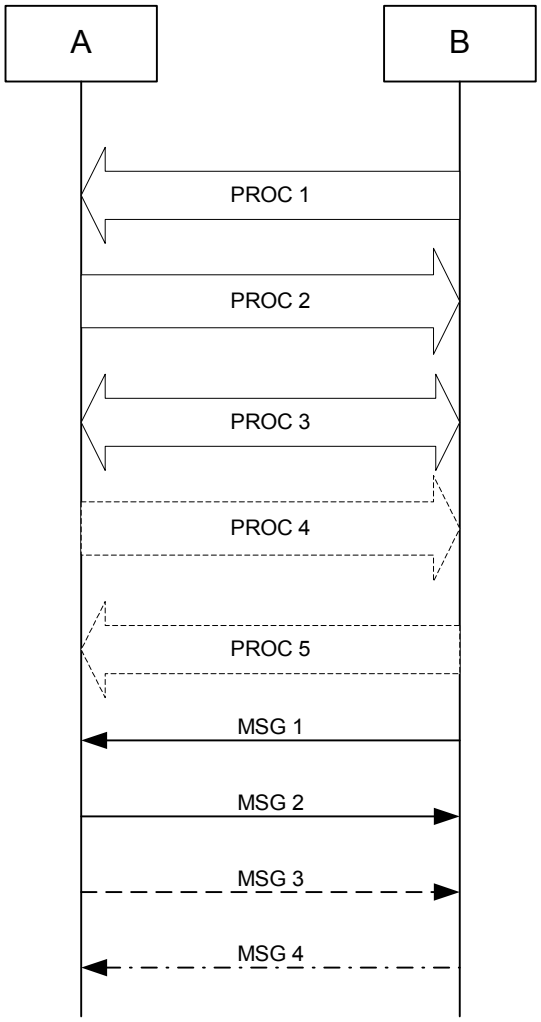


Figure B.1—Arrows used in signaling diagrams

In Figure B.1, the following cases are shown: PROC1 is a subprocedure initiated by B. PROC2 is a subprocedure initiated by A. PROC3 is a subprocedure where the initiating side is undefined (may be both A or B). Dashed arrows denote optional steps. PROC4 indicates an optional subprocedure initiated by A, and PROC5 indicates an optional subprocedure initiated by B.

MSG1 is a message sent from B to A. MSG2 is a message sent from A to B. MSG3 indicates an optional message from A to B, and MSG4 indicates a conditional message from B to A.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

B.2.3 Notation for timers and counters

Timers are introduced specific to this profile. To distinguish them from timers used in the IEEE 802.15.1-2005 protocol specifications and Bluetooth profiles, these timers are named in the following format: $T_{GAP}(nnn)$.

B.3 Profile overview

B.3.1 Profile stack

The main purpose of the GAP is to describe the use of the lower layers of the IEEE 802.15.1-2005 protocol stack (i.e., link controller and LMP). To describe security-related alternatives, higher layers [e.g., L2CAP, RFCOMM, and Object Exchange Protocol (OBEX)] are also included. See Figure B.2.

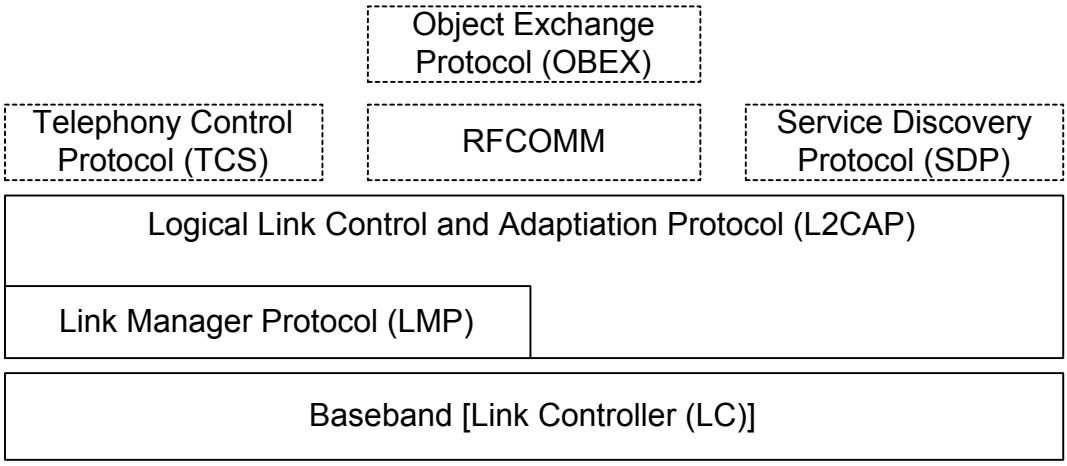


Figure B.2—Profile stack covered by this profile

B.3.2 Configurations and roles

In the profile in Figure B.3, the generic notation of the A-party (the paging device in case of link establishment or initiator in case of another procedure on an established link) and the B-party (paged device or acceptor) is used for the descriptions of the roles that the two devices involved in an IEEE 802.15.1-2005 communication can take. The A-party is the one that, for a given procedure, initiates the establishment of the physical link or initiates a transaction on an existing link.

This profile handles the procedures between two devices related to discovery and connecting (link and connection establishment) for the case where none of the two devices has any link established as well as the case where (at least) one device has a link established (possibly to a third device) before starting the described procedure.

The initiator and the acceptor generally operate the generic procedures according to this profile or another profile referring to this profile. If the acceptor operates according to several profiles simultaneously, this profile describes generic mechanisms for how to handle this.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

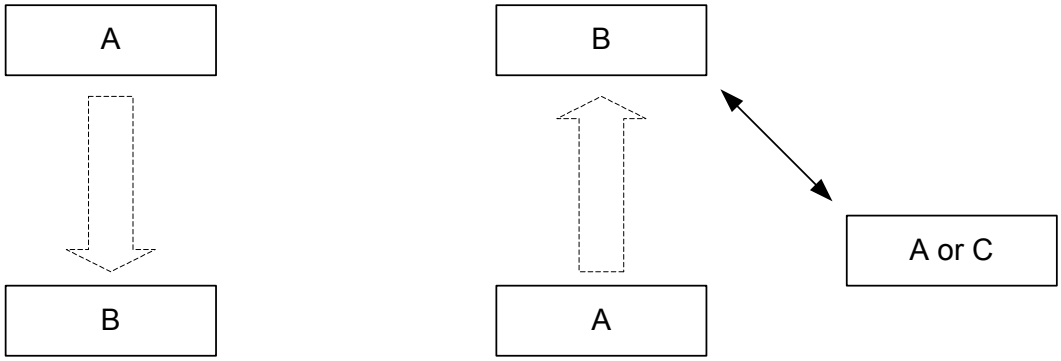


Figure B.3—Profile covering procedures initiated by one device (A) toward another device (B) that may or may not have existing IEEE 802.15.1-2005 link active

B.3.3 User requirements and scenarios

The IEEE 802.15.1-2005 user should in principle be able to connect a device to any other device. Even if the two connected devices do not share any common application, it should be possible for the user to find this out using basic IEEE 802.15.1-2005 capabilities. When the two devices do share the same application, but are from different manufacturers, the ability to connect them should not be blocked just because manufacturers choose to call basic IEEE 802.15.1-2005 capabilities by different names on the user interface level or implement basic procedures to be executed in different orders.

B.3.4 Profile fundamentals

The GAP performs the following fundamentals:

- States the requirements on names, values, and coding schemes used for names of parameters and procedures experienced on the user interface level.
- Defines modes of operation that are not service- or profile-specific, but that are generic and can be used by profiles referring to this profile and by devices implementing multiple profiles.
- Defines the general procedures that can be used for discovering identities, names, and basic capabilities of other devices that are in a mode where they can be discoverable. Only procedures where no channel or connection establishment is used are specified.
- Defines the general procedure for how to create bonds (i.e., dedicated exchange of link keys) between devices.
- Describes the general procedures that can be used for establishing connections to other devices that are in a mode that allows them to accept connections and service requests.

B.4 Modes

Conformance requirements related to modes defined in this clause are summarized in Table B.1.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table B.1—Conformance requirements related to modes defined in this clause

Procedure	Ref.	Support
Discoverability modes:	B.4.1	
Nondiscoverable mode		C1
Limited discoverable mode		O
General discoverable mode		O
Connectability modes:	B.4.2.2	
Nonconnectable mode		O
Connectable mode		M
Pairing modes:	B.9.2	
Nonpairable mode		O
Pairable mode		C2
C1: If limited discoverable mode is supported, nondiscoverable mode is mandatory; otherwise, optional. C2: If the bonding procedure is supported, support for pairable mode is mandatory; otherwise, optional.		

B.4.1 Discoverability modes

With respect to inquiry, a device shall be either in nondiscoverable mode or in a discoverable mode. (The device shall be in one, and only one, discoverability mode at a time.) The two discoverable modes defined here are called *limited discoverable mode* and *general discoverable mode*. Inquiry is defined in Clause 8.

When a device is in nondiscoverable mode, it does not respond to inquiry.

A device is said to be made discoverable, or set into a discoverable mode, when it is in limited discoverable mode or in general discoverable mode. Even when a device is made discoverable, it may be unable to respond to inquiry due to other BB activity. A device that does not respond to inquiry for any of these two reasons is called a *silent device*.

After being made discoverable, the device shall be discoverable for at least $T_{GAP}(103)$.

The speed of discovery is dependent on the configuration of the inquiry scan interval and inquiry scan type of the device. The host is able to configure these parameters based on trade-offs between power consumption, bandwidth, and the desired speed of discovery.

B.4.1.1 Nondiscoverable mode

When a device is in nondiscoverable mode, it shall never enter the INQUIRY_RESPONSE state.

B.4.1.2 Limited discoverable mode

The limited discoverable mode should be used by devices that need to be discoverable only for a limited period of time, during temporary conditions, or for a specific event. The purpose is to respond to a device that makes a limited inquiry [inquiry using the limited inquiry access code (LIAC)].

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

A device should not be in limited discoverable mode for more than $T_{GAP}(104)$. The scanning for the limited IAC can be done either in parallel or in sequence with the scanning of the GIAC. When in limited discoverable mode, one of the following options shall be used:

— *Parallel scanning.* When a device is in limited discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that interlaced inquiry scan be used.

  If, however, power consumption or bandwidth is important, but not critical, it is recommended that the device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and interlaced inquiry scan be used.

  When power consumption or bandwidth is critical, it is recommended that the device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and noninterlaced inquiry scan be used.

  In all cases, the device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC and the LIAC for at least $T_{GAP}(101)$.

  When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

— *Sequential scanning.* When a device is in limited discoverable mode, it shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$. It shall also enter the INQUIRY_SCAN state more often than once in $T_{GAP}(102)$ and scan for the LIAC for at least $T_{GAP}(101)$.

  If an inquiry message is received when in limited discoverable mode, the entry into the INQUIRY_RESPONSE state takes precedence over the next entries into INQUIRY_SCAN state until the inquiry response is completed.

When a device is in limited discoverable mode, it shall set bit 13 in the Major Service Class subfield of the Class of Device/Service field. The values are specified in Bluetooth Assigned Numbers [B1].

B.4.1.3 General discoverable mode

The general discoverable mode shall be used by devices that need to be discoverable continuously or for no specific condition. The purpose is to respond to a device that makes a general inquiry (inquiry using the GIAC).

When a device is in general discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that interlaced inquiry scan be used.

If, however, power consumption or bandwidth is important, but not critical, it is recommended that the device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and interlaced inquiry scan be used.

When power consumption or bandwidth is critical, it is recommended that the device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and noninterlaced inquiry scan be used.

In all cases, the device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$.

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

A device in general discoverable mode shall not respond to an LIAC inquiry.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

B.4.2 Connectability modes

With respect to paging, a device shall be either in nonconnectable mode or connectable mode. Paging is defined in Clause 8.

When a device is in nonconnectable mode, it does not respond to paging. When a device is in connectable mode, it responds to paging.

The speed of connections is dependent on the configuration of the page scan interval and page scan type of the device. The host is able to configure these parameters based on trade-offs between power consumption, bandwidth, and the desired speed of connection.

B.4.2.1 Nonconnectable mode

When a device is in nonconnectable mode, it shall never enter the PAGE_SCAN state.

B.4.2.2 Connectable mode

When a device is in connectable mode, it shall periodically enter the PAGE_SCAN state. The device makes page scan using the device address, BD_ADDR. Connection speed is a trade-off between

— Power consumption and available bandwidth and
— Speed.

The Bluetooth host is able to make these trade-offs using the page scan interval, page scan window, and interlaced scan parameters.

R0 page scanning should be used when connection speeds are critically important and when the paging device has a very good estimate of the clock. Under these conditions, it is possible for paging to complete within two times the page scan window. Because the page scan interval is equal to the page scan window, it is not possible for any other traffic to go over the Bluetooth link when using R0 page scanning. In R0 page scanning, it is not possible to use interlaced scan. R0 page scanning is the highest power consumption mode of operation.

When connection times are critical, but either the other device does not have an estimate of the clock or when the estimate is possibly out of date, it is better to use R1 page scanning with a very short page scan interval, $T_{GAP}(106)$, and interlaced scan. This configuration is also useful to achieve nearly the same connection speeds as R0 page scanning, but using less power consumption and leaving bandwidth available for other connections. Under these circumstances, it is possible for paging to complete within $T_{GAP}(106)$. In this case, the device shall page scan for at least $T_{GAP}(101)$.

When connection times are important, but not critical enough to sacrifice significant bandwidth and/or power consumption, it is recommended to use either $T_{GAP}(107)$ or $T_{GAP}(108)$ for the scanning interval. Using interlaced scan will reduce the connection time by half, but may use twice the power consumption. Under these circumstances, it is possible for paging to complete within one or two times the page scanning interval depending on whether interlaced scan is used. In this case, the device shall page scan for at least $T_{GAP}(101)$.

In all cases, the device shall enter the PAGE_SCAN state at least once in $T_{GAP}(102)$ and scan for at least $T_{GAP}(101)$.

The page scan interval, page scan window size, and scan type for the six scenarios are described in Table B.2.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table B.2—Page scan parameters for connection speed scenarios

Scenario	Page scan interval	Page scan window	Scan type
R0 (1.28 s)	$T_{GAP}(107)$	$T_{GAP}(107)$	Normal scan
Fast R1 (100 ms)	$T_{GAP}(106)$	$T_{GAP}(101)$	Interlaced scan
Medium R1 (1.28 s)	$T_{GAP}(107)$	$T_{GAP}(101)$	Interlaced scan
Slow R1 (1.28 s)	$T_{GAP}(107)$	$T_{GAP}(101)$	Normal scan
Fast R2 (2.56 s)	$T_{GAP}(108)$	$T_{GAP}(101)$	Interlaced scan
Slow R2 (2.56 s)	$T_{GAP}(108)$	$T_{GAP}(101)$	Normal scan

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the connection time.

B.4.3 Pairing modes

With respect to pairing, a device shall be either in nonpairable mode or in pairable mode. In pairable mode, the device accepts pairing, i.e., creation of bonds, initiated by the remote device; and in nonpairable mode, it does not. Pairing is defined in Clause 8 and Clause 9.

B.4.3.1 Nonpairable mode

When a device is in nonpairable mode, it shall respond to a received LMP_in_rand PDU with an LMP_not_accepted PDU with the reason *pairing not allowed*.

B.4.3.2 Pairable mode

When a device is in pairable mode, it shall respond to a received LMP_in_rand PDU with an LMP_accepted PDU (or with an LMP_in_rand PDU if it has a fixed PIN).

B.5 Security aspects

Conformance requirements related to the generic authentication procedure and the security modes defined in this clause are summarized in Table B.3.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table B.3—Conformance requirements related to the generic authentication procedure and the security modes defined in this clause

	Procedure	Ref.	Support
1	Authentication	B.5.1	C1
2	Security modes	B.5.2	
	Security mode 1		O
	Security mode 2		C2
	Security mode 3		C2
C1: If security mode 1 is the only security mode that is supported, support for authentication is optional; otherwise, mandatory. (Note that support for LMP authentication and LMP pairing is mandatory according to Clause 9, independent of the security mode that is used.) C2: If secure communication is supported, then support for at least one of security mode 2 or security mode 3 is mandatory.			

B.5.1 Authentication

The generic authentication procedure describes how the LMP authentication and LMP pairing procedures are used when authentication is initiated by one device toward another, depending on if a link key exists or not and if pairing is allowed or not.

B.5.1.1 Procedure

The generic authentication procedure is defined in Figure B.4.

The device that initiates authentication has to be in security mode 2 or in security mode 3.

B.5.2 Security modes

The flow chart in Figure B.5 describes where in the channel establishment procedure's initiation of authentication takes place, depending on the security mode in which the device is operating.

When authentication is initiated toward a device, it shall act according to Clause 9 and the current pairing mode, independent of the security mode in which it is operating.

B.5.2.1 Security mode 1 (nonsecure)

When a device is in security mode 1, it shall never initiate any security procedure (i.e., it shall never send LMP_au_rand, LMP_in_rand, or LMP_encryption_mode_req PDUs).

B.5.2.2 Security mode 2 (service-level enforced security)

When a device is in security mode 2, it shall not initiate any security procedure before a channel establishment request (L2CAP_ConnectReq) has been received or a channel establishment procedure has been initiated by itself. Whether a security procedure is initiated depends on the security requirements of the requested channel or service.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

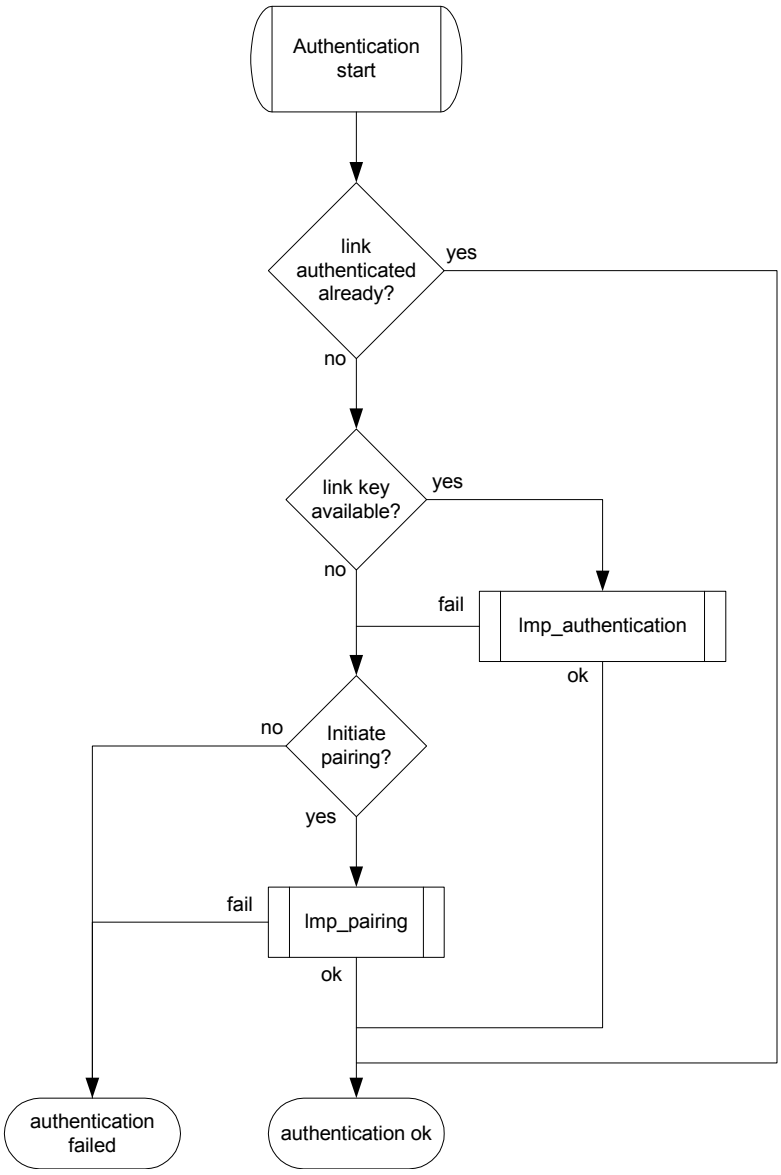


Figure B.4—Definition of the generic authentication procedure

A device in security mode 2 should classify the security requirements of its services using at least the following attributes:

— Authorization required
— Authentication required
— Encryption required

NOTE—Security mode 1 can be considered (at least from a remote device's point of view) as a special case of security mode 2 where no service has registered any security requirements.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

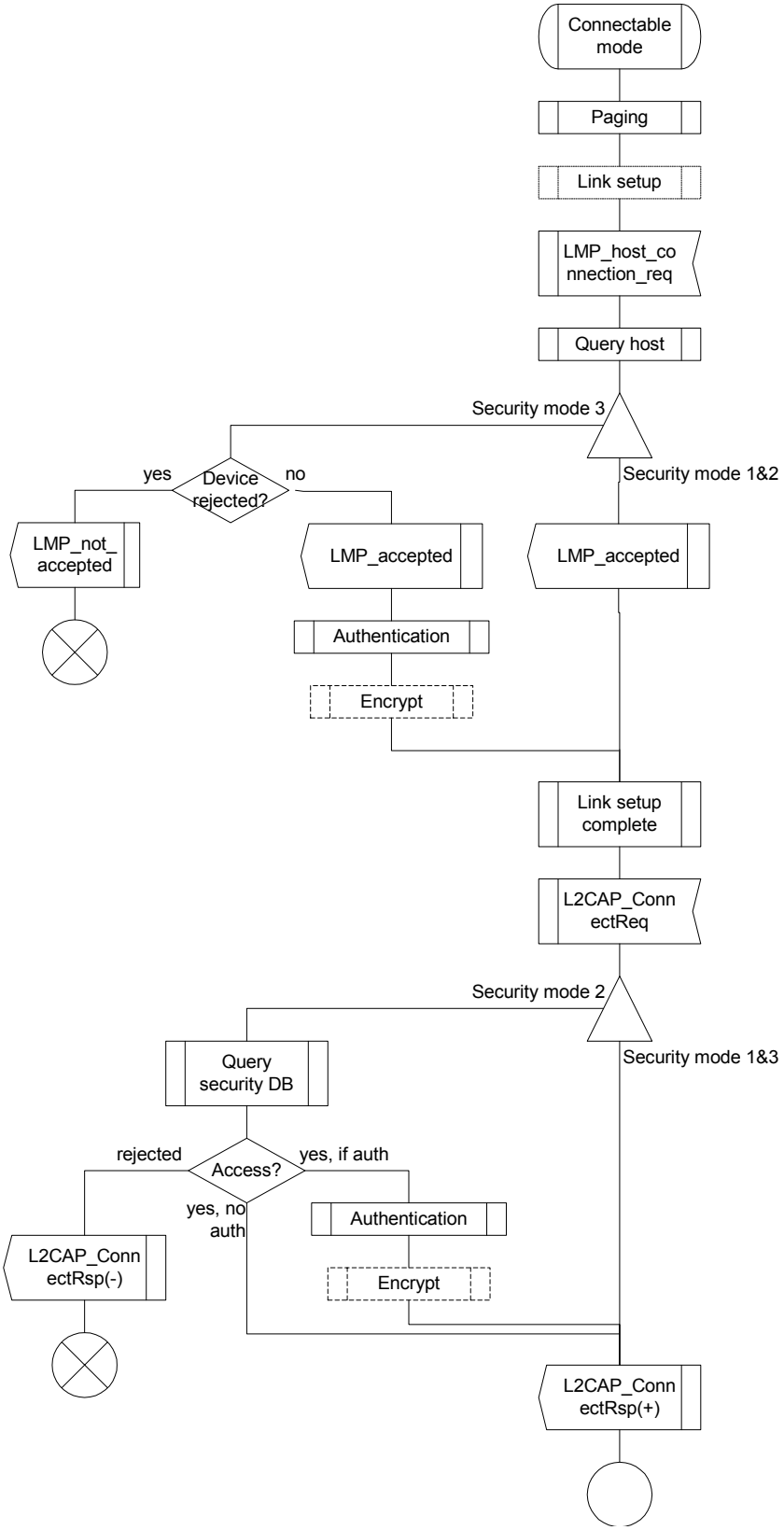


Figure B.5—Illustration of channel establishment using different security modes

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

B.5.2.3 Security modes 3 (link level enforced security)

When a device is in security mode 3, it shall initiate security procedures before it sends an LMP_link_setup_ complete PDU. (The behavior of a device in security mode 3 is as described in Clause 9.)

A device in security mode 3 may reject the host connection request (respond with an LMP_not_accepted PDU to the LMP_host_connection_req PDU) based on settings in the host (e.g., only communication with prepaired devices allowed).

B.6 Idle mode procedures

The inquiry and discovery procedures described here are applicable only to the device that initiates them (device A). The requirements on the behavior of device B are according to the modes specified in B.4 and to Clause 9. The conformance requirements related to idle mode procedures defined in this clause are summarized in Table B.4.

Table B.4—Conformance requirements related to idle mode procedures defined in this clause

	Procedure	Ref.	Support
1	General inquiry	B.6.1	C1
2	Limited inquiry	B.6.2	C1
3	Name discovery	B.6.3	O
4	Device discovery	B.6.3.3	O
5	Bonding	B.6.4	O

C1: If initiation of bonding is supported, support for at least one inquiry procedure is mandatory; otherwise, optional.
NOTE—Support for LMP pairing is mandatory Clause 9.

B.6.1 General inquiry

The purpose of the general inquiry procedure is to provide the initiator with the device address, clock, class of device, and used page scan mode of general discoverable devices (i.e., devices that are in range with regard to the initiator and are set to scan for inquiry messages with the GIAC). Also devices in limited discoverable mode will be discovered using general inquiry.

The general inquiry should be used by devices that need to discover devices that are made discoverable continuously or for no specific condition. See Figure B.6.

When general inquiry is initiated by a device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The device shall perform inquiry using the GIAC.

In order to receive inquiry response, the remote devices in range have to be made discoverable (limited or general).

NOTE—All discoverable devices are discovered using general inquiry, independent of the discoverable mode in which they are operating.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

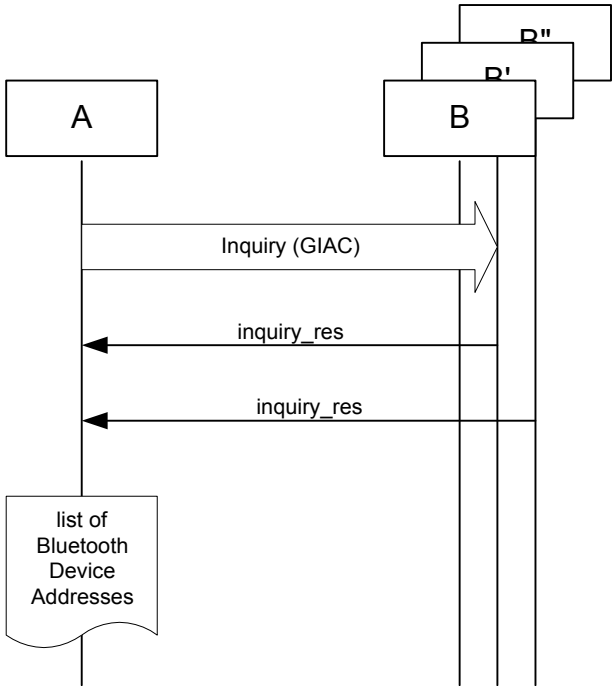


Figure B.6—General inquiry, where device B is in nondiscoverable mode, device B' is in limited discoverable mode, and device B" is in general discoverable mode

B.6.2 Limited inquiry

The purpose of the limited inquiry procedure is to provide the initiator with the device address, clock, class of device, and used page scan mode of limited discoverable devices. The latter devices are devices that are in range with regard to the initiator and may be set to scan for inquiry messages with the limited IAC, in addition to scanning for inquiry messages with the GIAC.

The limited inquiry should be used by devices that need to discover devices that are made discoverable only for a limited period of time, during temporary conditions, or for a specific event. Since it is not guaranteed that the discoverable device scans for the LIAC, the initiating device may choose any inquiry procedure (general or limited). Even if the remote device that is to be discovered is expected to be made limited-discoverable (e.g., when a dedicated bonding is to be performed), the limited inquiry should be done in sequence with a general inquiry in such a way that both inquiries are completed within the time the remote device is limited-discoverable, i.e., at least $T_{GAP}(103)$. See Figure B.7.

When limited inquiry is initiated by a device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The device shall perform inquiry using the LIAC.

In order to receive inquiry response, the remote devices in range has to be made limited-discoverable.

NOTE—Only limited-discoverable devices can be discovered using limited inquiry.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

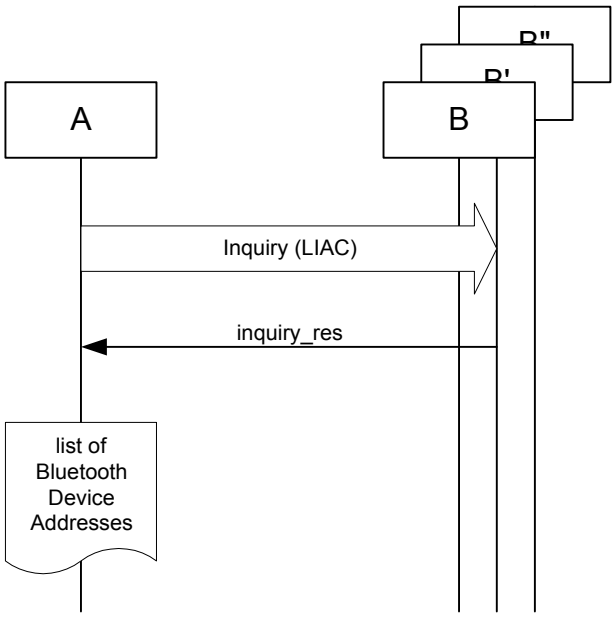


Figure B.7—Limited inquiry, where device B is in nondiscoverable mode, device B' is in limited discoverable mode, and device B" is in general discoverable mode

B.6.3 Name discovery

The purpose of name discovery is to provide the initiator with the device name of connectable devices (i.e., devices in range that will respond to paging).

B.6.3.1 Name request

Name request is the procedure for retrieving the device name from a connectable device. It is not necessary to perform the full link establishment procedure (see B.7.1) in order to simply get the name of another device. See Figure B.8.

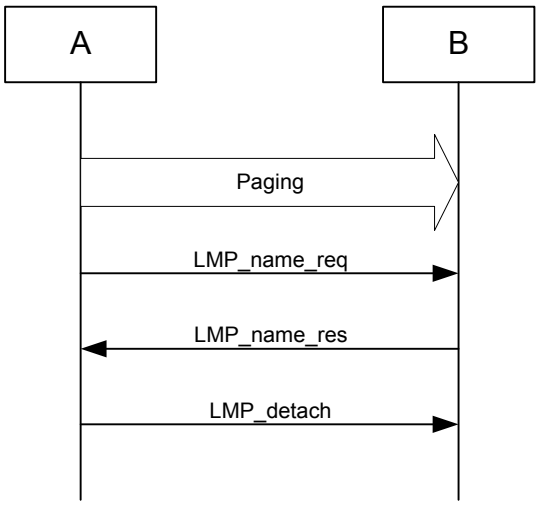


Figure B.8—Name request procedure

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

B.6.3.2 Name discovery

Name discovery is the procedure for retrieving the device name from connectable devices by performing name request toward known devices (i.e., devices for which the device addresses are available). See Figure B.9.

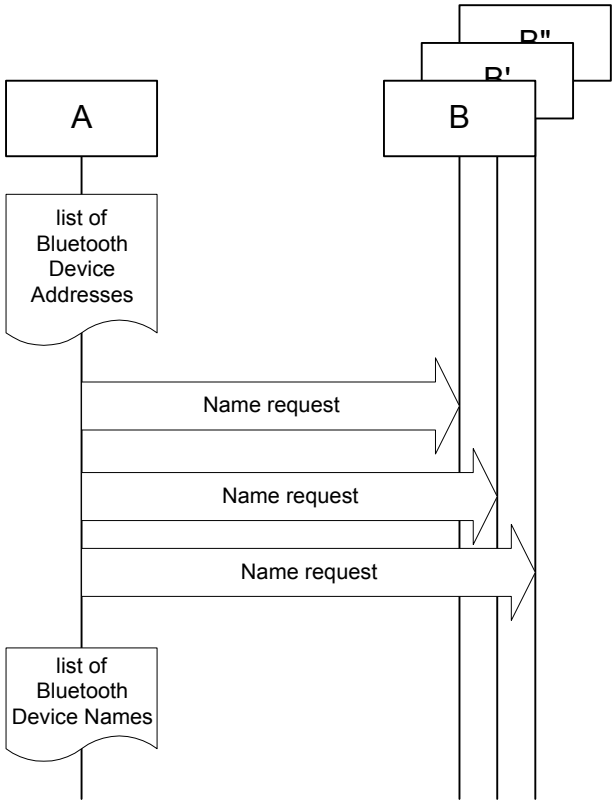


Figure B.9—Name discovery procedure

In the name request procedure, the initiator will use the DAC of the remote device as retrieved immediately beforehand, normally through an inquiry procedure.

B.6.3.3 Device discovery

The purpose of device discovery is to provide the initiator with the device address, clock, class of device, used page scan mode, and device name of discoverable devices.

During the device discovery procedure, first an inquiry (either general or limited) is performed, and then name discovery is done toward some or all of the devices that responded to the inquiry. See Figure B.10.

Conditions for both inquiry (general or limited) and name discovery must be fulfilled (i.e., devices discovered during device discovery must be both discoverable and connectable).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

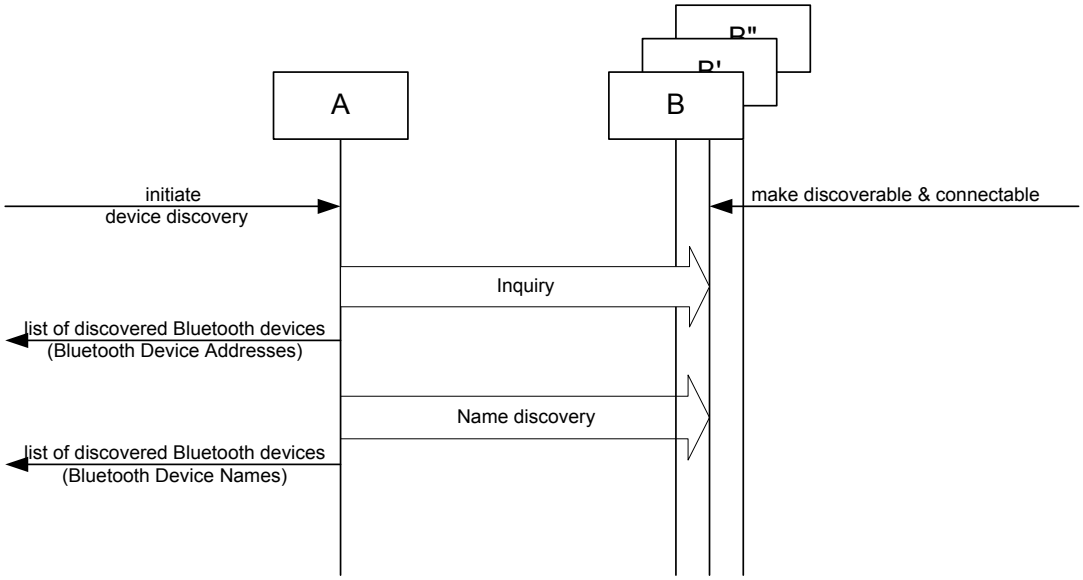


Figure B.10—Device discovery procedure

B.6.4 Bonding

The purpose of bonding is to create a relation between two devices based on a common link key (a bond). The link key is created and exchanged (pairing) during the bonding procedure and is expected to be stored by both devices to be used for future authentication.

In addition to pairing, the bonding procedure can involve higher layer initialization procedures.

Two aspects of the bonding procedure are described here:

— Dedicated bonding is what is done when the two devices are explicitly set to perform only a creation and exchange of a common link key.
— General bonding is included to indicate that the framework for the dedicated bonding procedure is the same as found in the normal channel and connection establishment procedures. This means that pairing may be performed successfully if device A has initiated bonding while device B is in its normal connectable and security modes.

The main difference with bonding, as compared to a pairing done during link or channel establishment, is that for bonding it is the paging device (A) that must initiate the authentication.

B.6.4.1 General bonding

Figure B.11 shows a description of general bonding, where the link establishment procedure is executed under specific conditions on both devices, followed by an optional higher layer initalization process.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

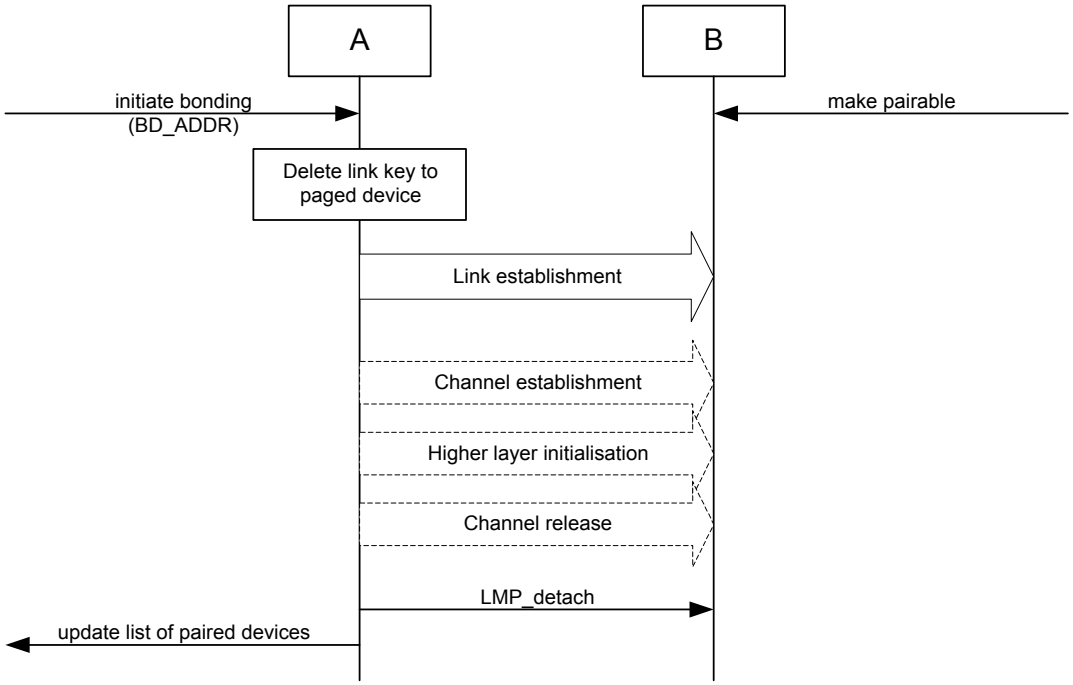


Figure B.11—General bonding, where the link establishment procedure is executed under specific conditions on both devices, followed by an optional higher layer initalization process

B.6.4.2 Dedicated bonding

Figure B.12 shows a description of dedicated bonding, where the purpose of the procedure is only to create and exchange a link key between two devices.

Before bonding can be initiated, the initiating device (A) must know the DAC of the device to pair with. This is normally done by first performing device discovery. A device that can initiate bonding (A) should use limited inquiry, and a device that accepts bonding (B) should support the limited discoverable mode.

Bonding is in principle the same as link establishment with the following conditions:

— The paged device (B) shall be set into pairable mode. The paging device (A) is assumed to allow pairing since it has initiated the bonding procedure.
— The paging device (the initiator of the bonding procedure, A) shall initiate authentication.
— Before initiating the authentication part of the bonding procedure, the paging device should delete any link key corresponding to a previous bonding with the paged device.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

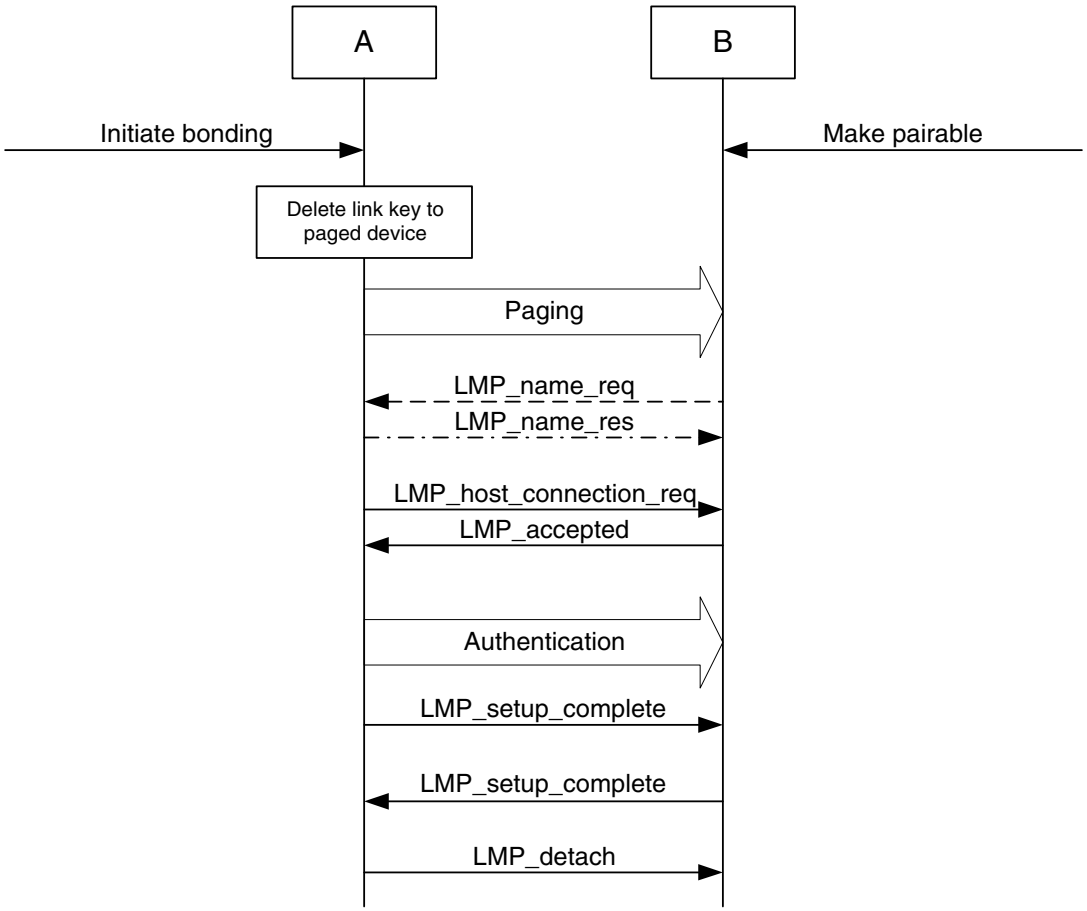


Figure B.12—Dedicated bonding, when the purpose of the procedure is only to create and exchange a link key between two devices

B.7 Establishment procedures

The establishment procedures defined in this clause do not include any discovery part and are summarized in Table B.5.

Table B.5—Establishment procedures defined in this clause

	Procedure	Ref.	Support in A	Support in B
1	Link establishment	B.7.1	M	M
2	Channel establishment	B.7.2	O	M
3	Connection establishment	B.7.3	O	O

Before establishment procedures are initiated, the following information, provided during device discovery (in the FHS packet of the inquiry response or in the response to a name request), has to be available in the initiating device:

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

— The device address (BD_ADDR) from which the DAC is generated
— The system clock of the remote device
— The page scan mode used by the remote device

Additional information provided during device discovery that is useful for making the decision to initiate an establishment procedure is the following:

— The class of device
— The device name

B.7.1 Link establishment

The purpose of the link establishment procedure is to establish a physical link (of ACL type) between two devices using procedures from Clause 8 and Clause 9.

B.7.1.1 Device B in security mode 1 or 2

Figure B.13 shows the link establishment procedure when the paging device (A) is in security mode 3 and the paged device (B) is in security mode 1 or 2. During link establishment, the paging device cannot distinguish if the paged device is in security mode 1 or 2.

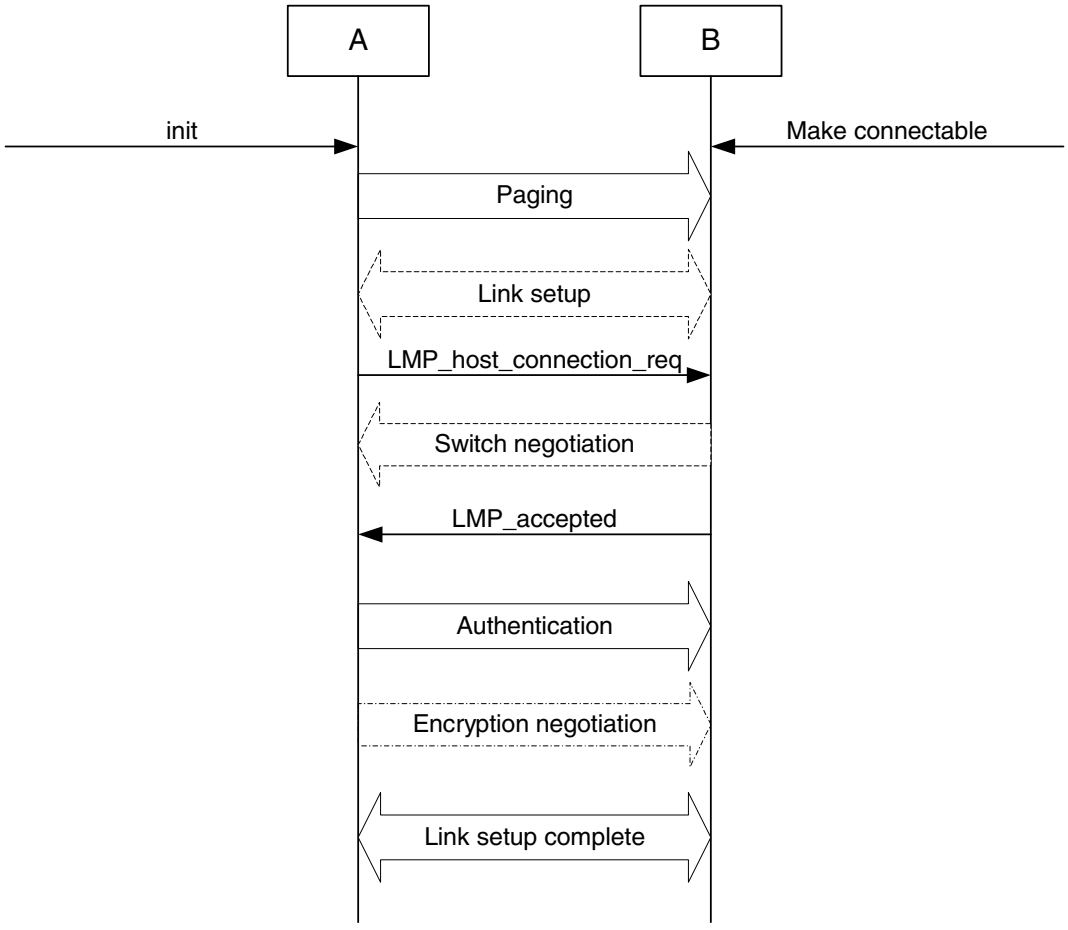


Figure B.13—Link establishment procedure when the paging device (A) is in security mode 3 and the paged device (B) is in security mode 1 or 2

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

B.7.1.2 Device B in security mode 3

Figure B.14 shows the link establishment procedure when both the paging device (A) and the paged device (B) are in security mode 3.

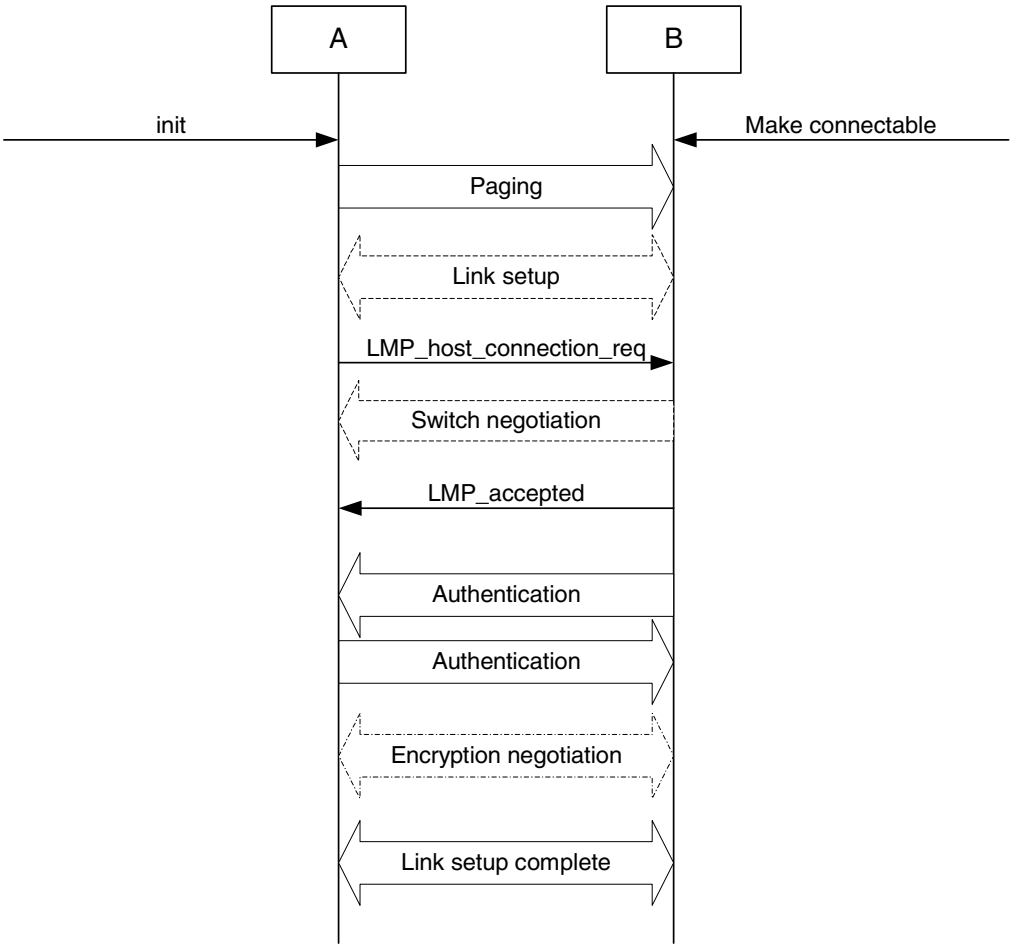


Figure B.14—Link establishment procedure when both the paging device (A) and the paged device (B) are in security mode 3

The paging procedure shall be according to Clause 8, and the paging device should use the DAC and page mode received through a previous inquiry. When paging is completed, a physical link between the two devices is established.

If role switching is needed (normally it is the paged device that has an interest in changing the master/slave roles), it should be done as early as possible after the physical link is established. If the paging device does not accept the switch, the paged device has to consider whether to keep the physical link.

Both devices may perform link setup (using LMP procedures that require no interaction with the host on the remote side). Optional LMP features can be used after having confirmed (using LMP_feature_req) that the other device supports the feature.

When the paging device needs to go beyond the link setup phase, it issues a request to be connected to the host of the remote device. If the paged device is in security mode 3, this is the trigger for initiating authentication.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

The paging device shall send an LMP_host_connection_req PDU during link establishment (i.e., before channel establishment) and may initiate authentication only after having sent an LMP_host_connection_ request PDU.

After an authentication has been performed, any of the devices can initiate encryption.

Further link configuration may take place after the LMP_host_connection_req PDU. When both devices are satisfied, they send LMP_setup_complete PDUs.

Link establishment is completed when both devices have sent LMP_setup_complete PDUs.

B.7.2 Channel establishment

The purpose of the channel establishment procedure is to establish a Bluetooth channel (a logical link) between two devices using procedures specified in Clause 14.

B.7.2.1 Device B in security mode 2

Figure B.15 shows the channel establishment procedure when the initiator (device A) is in security mode 3 and the acceptor (device B) is in security mode 2.

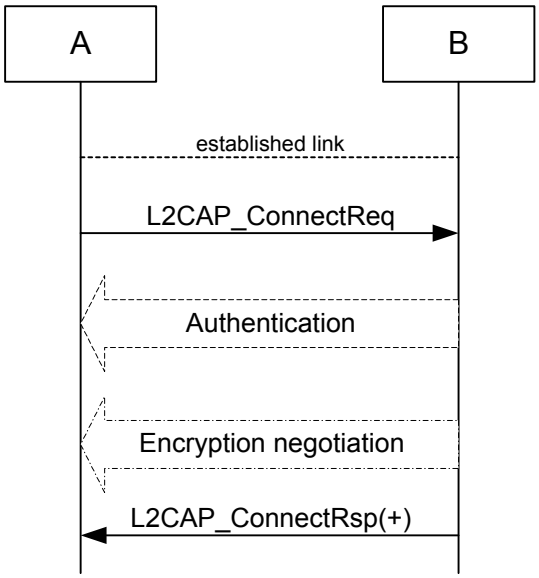


Figure B.15—Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2

B.7.2.2 Device B in security mode 1 or 3

Figure B.16 shows the channel establishment procedure when the initiator (device A) is in security mode 3 and the acceptor (device B) is in security mode 1 or 3. During channel establishment, the initiator cannot distinguish if the acceptor is in security mode 1 or 3.

Channel establishment starts after link establishment is completed when the initiator sends a channel establishment request (L2CAP_ConnectReq).

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

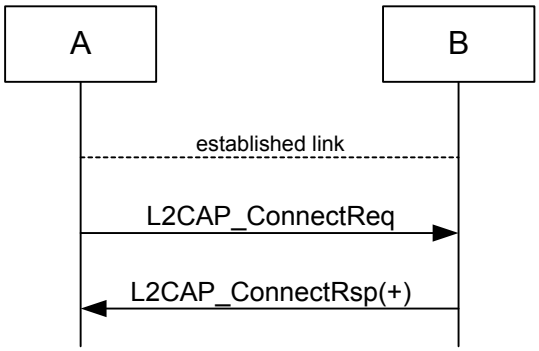


Figure B.16—Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3

Depending on security mode, security procedures may take place after the channel establishment has been initiated.

Channel establishment is completed when the acceptor responds to the channel establishment request (with a positive L2CAP_ConnectRsp).

B.7.3 Connection establishment

The purpose of the connection establishment procedure is to establish a connection between applications on two devices.

B.7.3.1 Device B in security mode 2

Figure B.17 shows the connection establishment procedure when the initiator (device A) is in security mode 3 and the acceptor (device B) is in security mode 2.

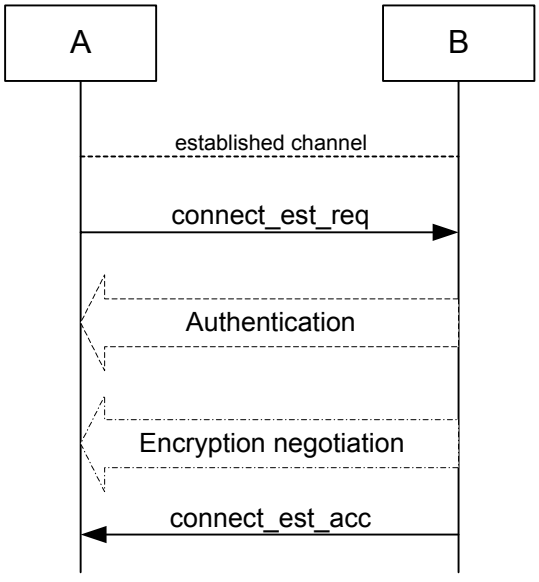


Figure B.17—Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

B.7.3.2 Device B in security mode 1 or 3

Figure B.18 shows the connection establishment procedure when the initiator (device A) is in security mode 3 and the acceptor (device B) is in security mode 1 or 3. During connection establishment, the initiator cannot distinguish if the acceptor is in security mode 1 or 3.

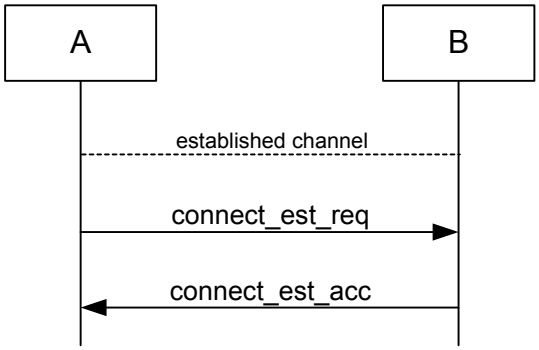


Figure B.18—Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3

Connection establishment starts after channel establishment is completed, when the initiator sends a connection establishment request ("connect_est_req" is application protocol dependent). This request may be a TCS SETUP message, in the case of a cordless telephony profile for a Bluetooth telephony application, or an initialization of RFCOMM and establishment of DLC, in the case of a serial port profile for a serial port-based application (although neither TCS or RFCOMM use the term *connection* for this).

Connection establishment is completed when the acceptor accepts the connection establishment request ("connect_est_acc" is application protocol dependent).

B.7.3.3 Establishment of additional connection

When a device has established one connection with another device, it may be available for establishment of the following additional connections:

— A second connection on the same channel, and/or
— A second channel on the same link, and/or
— A second physical link.

If the new establishment procedure is to be toward the same device, the security part of the establishment depends on the security modes used. If the new establishment procedure is to be toward a new remote device, the device should behave according to active modes independent of the fact that it already has another physical link established (unless allowed coincident radio and BB events have to be handled).

B.8 Timers and constants

The timers in Table B.6 are required by the GAP.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

Table B.6—Defined GAP timers

Timer name	Recommended value	Description	Comment
$T_{GAP}(100)$	10.24 s	Normal time span that a device performs inquiry.	Used during inquiry and device discovery.
$T_{GAP}(101)$	10.625 ms	Minimum time in INQUIRY_SCAN.	A discoverable device enters INQUIRY_SCAN for at least $T_{GAP}(101)$ every $T_{GAP}(102)$.
$T_{GAP}(102)$	2.56 s	Maximum time between repeated INQUIRY_SCAN enterings.	Maximum value of the inquiry scan interval $T_{inquiry\ scan}$.
$T_{GAP}(103)$	30.72 s	A device shall not be in a discoverable mode less than $T_{GAP}(103)$.	Minimum time to be discoverable.
$T_{GAP}(104)$	1 min	A device should not be in limited discoverable mode more than $T_{GAP}(104)$.	Recommended upper limit.
$T_{GAP}(105)$	100 ms	Maximum time between INQUIRY_SCAN enterings	Recommended value
$T_{GAP}(106)$	100 ms	Maximum time between PAGE_SCAN enterings	Recommended value
$T_{GAP}(107)$	1.28 s	Maximum time between PAGE_SCAN enterings (R1 page scan)	Recommended value
$T_{GAP}(108)$	2.56 s	Maximum time between PAGE_SCAN enterings (R2 page scan)	Recommended value

B.9 Information flows of related procedures

B.9.1 LMP authentication

The specification of authentication on link level is found in Clause 9. See also Figure B.19.

The secret key used here is an already exchanged link key.

B.9.2 LMP pairing

The specification of pairing on link level is found in Clause 9. See also Figure B.20.

The PIN used here is PN_{BB}.

The create link key procedure is described in 9.3.2.2.4 and 13.3.2. In case the link key is based on a combination key, a mutual authentication takes place and shall be performed irrespective of current security mode.

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

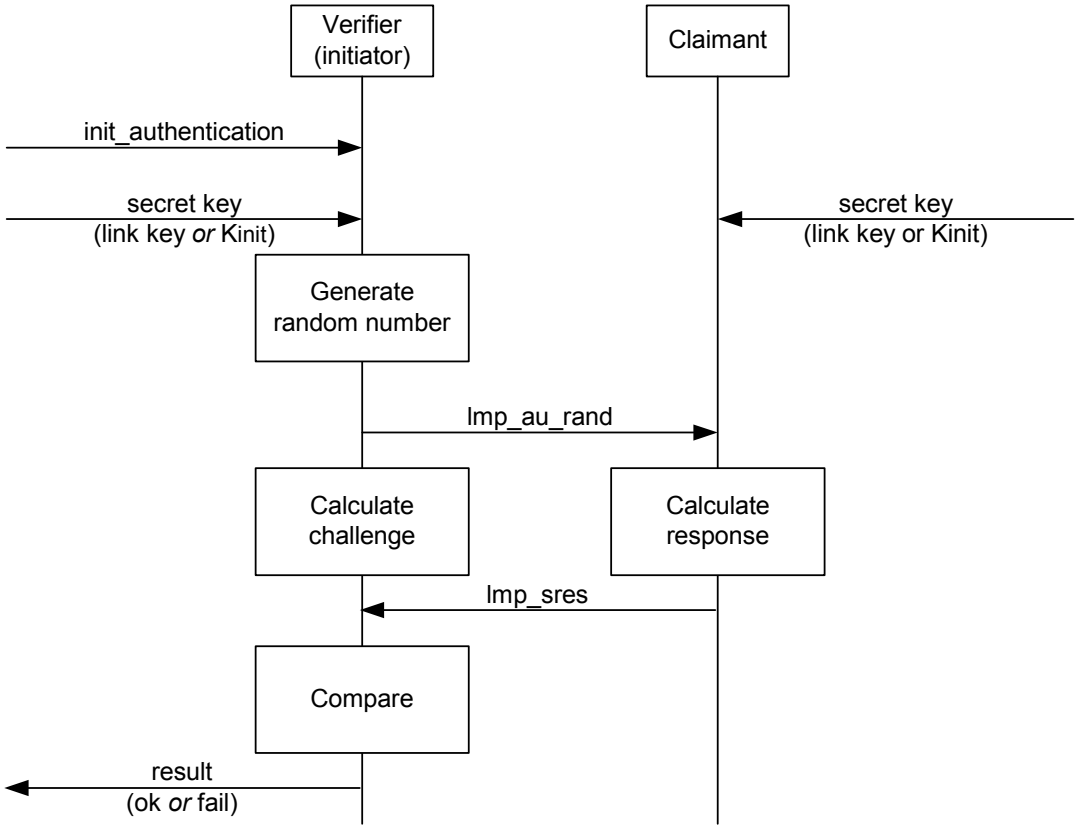


Figure B.19—LMP authentication as defined by Clause 9

B.9.3 Service discovery (SD)

The Service Discovery Protocol (SDP) (see Bluetooth Core Specification) specifies what PDUs are used over the air to inquire about services and service attributes. The procedures for discovery of supported services and capabilities using the SDP are described in the service discovery application profile (SDAP). Figure B.21 simply provides an example.

Copyright © 2005 IEEE. All rights reserved.

Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.

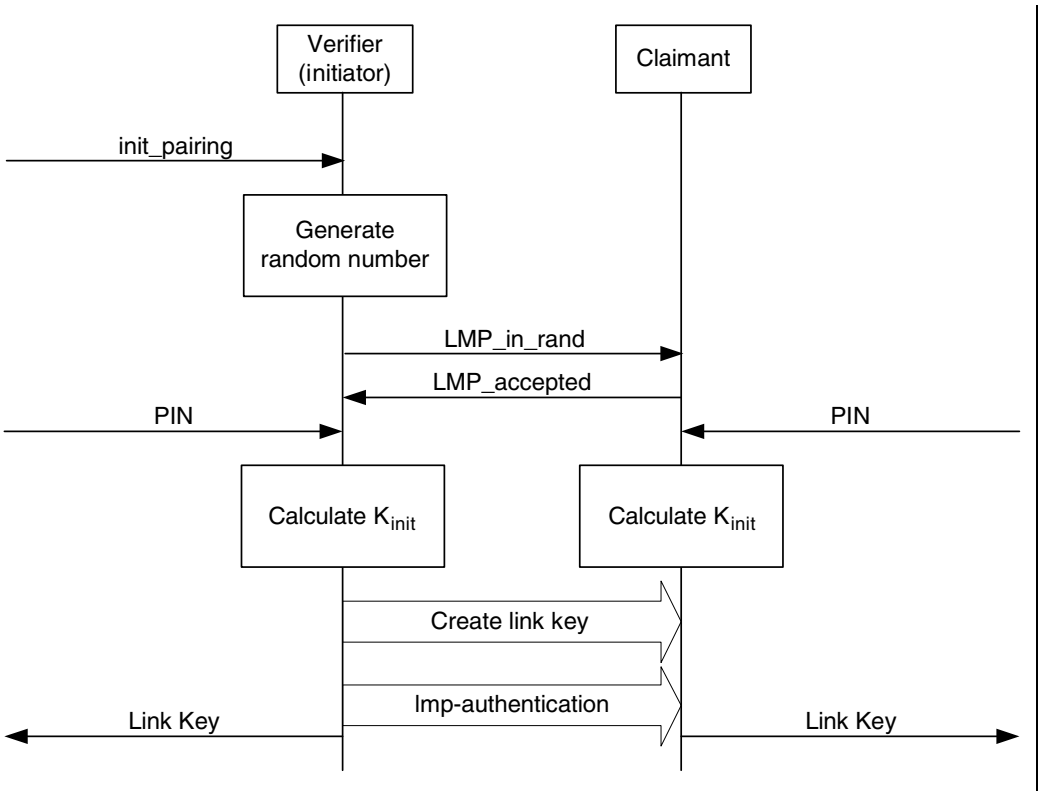


Figure B.20—LMP-pairing as defined in Clause 9

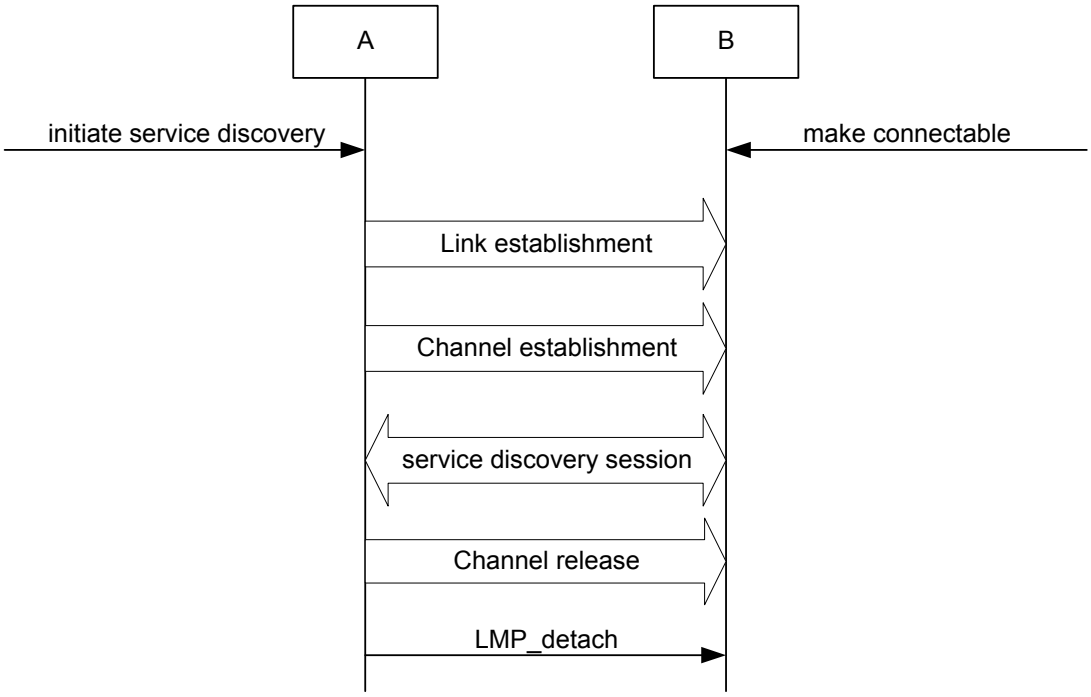


Figure B.21—SD procedure

Copyright © 2005 IEEE. All rights reserved.
Authorized licensed use limited to: University of Southern California. Downloaded on April 16,2025 at 16:39:28 UTC from IEEE Xplore. Restrictions apply.